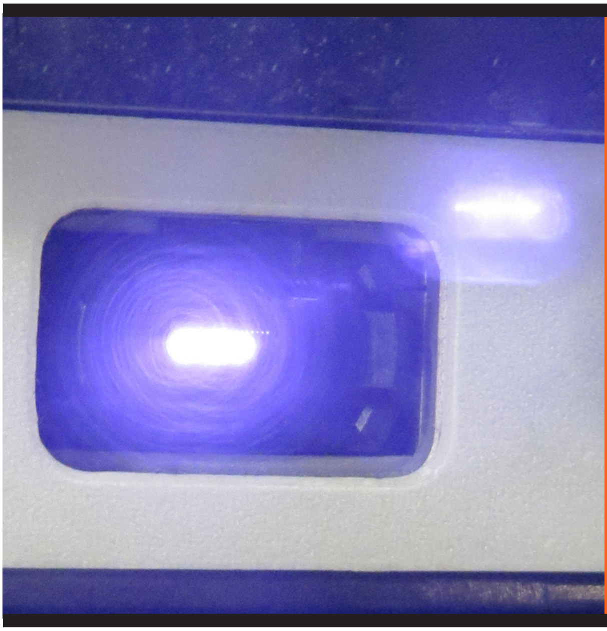

Towards Terabit Carrier Ethernet and Energy Efficient Optical Transport Networks

Anders Rasmussen
Ph. D. Thesis
October 2013

DTU Fotonik
Department of Photonics Engineering

Towards Terabit Carrier Ethernet and Energy Efficient Optical Transport Networks

Ph. D. thesis

Anders Rasmussen

DTU Fotonik,
Department of Photonics Engineering
Technical University of Denmark
October 2013

Supervisors
Michael S. Berger
Sarah Ruepp

Preface

This thesis represents the research carried out during my Ph. D studies from May 2010 to October 2013 at DTU Fotonik, Department of Photonics Engineering, Technical University of Denmark. The project has been supervised by Associate Professors Michael S. Berger and Sarah Ruepp and is partially funded by the Danish Advanced Technology Foundation (Højteknologifonden). The thesis is entitled "Towards Terabit Carrier Ethernet and Energy Efficient Optical Transport Networks" and investigates how future network elements can be scaled towards terabit-per-second capacities.

Abstract

This thesis focuses on the challenges of scaling current network node technology to support connection speeds of 100Gbps and beyond. Out of the many exiting aspects of reaching this goal, the main scope of this thesis is to investigate packet processing (address lookup and scheduling), forward error correction and energy efficiency.

Scheduling and address lookup are key functions and potential bottle necks in high speed network nodes, as the minimum packet/frame sizes in both the popular Ethernet protocol, as well as the Internet Protocol (IP) still remains constant (84B and 40B, respectively). Therefore, in order to support a single 100 Gigabit Ethernet link, the routing mechanism must be able to support address lookup and output scheduling of over 148 million packets per second (pps) leaving only a few nanoseconds for each packet. With the emerging standards for 400Gbps (400GE and OTU5) and discussions on how best to exceed the 1Tbps boundary, the packet processing rate requirements for future network nodes are likely to increase even further in the coming years. Hence, there lies a tremendous task in expanding and optimizing current technology and methodology to support these increasing requirements.

Forward Error Correction (FEC) is already a standard component of the Optical Transport Network (OTN) protocol as a means of increasing the bitrate-length product of optical links. However, the requirements for higher bitrates also drive a requirement for higher spectral efficiency in order to squeeze more traffic onto the existing physical transmission systems. To do this, while keeping the bit error rate (BER) below acceptable levels, more advanced FEC schemes are required. This is a challenge: Not only do we need to increase the processing speed of the FEC to handle the higher throughputs. The more advanced schemes also require more complex calculations to process each bit. This thesis will investigate how both the standard OTN FEC as well as more advanced FEC schemes can be implemented for 100G and above operation.

As the networks are expanded to run at increasingly higher speeds, an unfortunate by-product is higher energy consumption. While advances in the physical hardware production (e.g. better chip production techniques) somewhat reduces the problem, it is imperative to think energy efficiency into the systems from the early design stage to the actual implementation and operation. Similar to the now common practice within micro processors, recent research aims at dynamically balancing performance and energy consumption of optical communication systems based on the immediate capacity demands. This thesis will describe various ways of achieving this dynamic capacity/energy trade off, with special emphasis

on the Celtic project "Elastic Optical Networks" (EO-Net) and on adaptive forward error correction in particular.

Resumé

Denne afhandling omhandler udfordringerne i at skalere nuværende netværksnodeteknologi til at understøtte hastigheder på 100 gigabit per sekund og derover. Ud af de mange aspekter, som indgår i at nå dette mål, er fokus i denne afhandling lagt på pakkeprocessering (adresse opslag og schedulering), fremadrettet fejlkorrektion og energieffektivitet.

Schedulering og Adresse opslag er to nøglefunktioner og potentielle flaskehalse i højhastighedsnetværksnoder, da den mindste ramme/pakke størrelse i både den populære Ethernet protokol, samt i Internet Protokollen (IP), er holdt konstant (hhv. 84B og 40B). For at kunne understøtte en 100 Gigabit Ethernet forbindelse, skal rutningsmekanismerne derfor være i stand til at udføre adresseopslag og output schedulering af over 148 millioner pakker per sekund (pps), hvilket kun efterlader nogle få nanosekunder til at behandle hver pakke. Med de kommende standarder for 400Gbps (400GE og OTU5) og en igangværende diskussion om, hvordan man bedst overskrider 1 terabit/s grænsen, vil pakkeprocesseringshastigheden for fremtidige netværks-knudepunkter med stor sandsynlighed stige endnu mere i de kommende år. Der ligger derfor en gevaldig opgave i at udvide og optimere nuværende teknologi og metoder til at understøtte disse stigende krav.

Fremadrettet fejlkorrektion (Forward Error Correction (FEC)) er allerede en standardkomponent i Optical Transport Network (OTN) protokollen, og tjener det formål at forøge bitrate-længde produktet på optiske forbindelser. Kravet om højere bitrater skaber også forøgede krav til spektraleffektiviteten for at klemme endnu mere kapacitet ud af eksisterende fysiske transmissionssystemer. For at gøre dette og stadig holde bitfejlsandsynligheden (BER) under et acceptabelt niveau, er det nødvendigt at benytte mere avancerede FEC metoder. Dette er en stor udfordring: Ud over at FEC systemerne skal køre hurtigere for at understøtte højere datarater, vil de mere komplicerede FEC metoder også kræve mere komplicerede udregninger for de enkelte bits. I denne afhandling vil det blive undersøgt, hvordan FEC systemet fra OTN, såvel som mere avancerede FEC metoder, kan implementeres til at køre med linierater på 100Gbps og derover.

En uheldig bivirkning ved de stadig større datahastigheder er øget energiforbrug. Til trods for at forbedringer af den fysiske hardwareproduktion til dels modvirker denne udvikling, er det yderst nødvendigt at tænke energieffektivitet ind i systemerne lige fra den tidlige designfase og indtil systemerne implementeres og tages i brug. I tråd med den nu gængse praksis indenfor mikroprocessorer, stiler nylig forskning efter dynamisk at afbalancere ydelse og energiforbrug i optiske kommunikationssystemer ud fra de øjeblikkelige kapacitetskrav. Denne

afhandling vil beskrive forskellige metoder til at opnå dynamisk afvejning mellem kapacitets- og energihensyn, med fokus på Celtic-projektet ”Elatic Optical Networks” (EO-Net) og i særlig grad på adaptiv fremadrettet fejlkorrektion.

Acknowledgements

The contents of this Ph. D. thesis would not have existed without the help, support and good company of a number of people.

Firstly, I would like to thank my supervisors Michael Berger and Sarah Ruepp for their invaluable guidance and support during the last few years. Whenever I needed help with an article, structural guidance for my Ph. D study, or just to brainstorm an idea, your doors were always open.

Secondly, I would like to thank the co-authors of my research papers for their important contributions to this thesis. Without your input, this work would be incomplete.

I would also like to thank Hao Yu, Ying Yan, José Soler, Anna Manolova, Henrik Wessing, Michael Berger and Sarah Ruepp for proof read and commenting on this thesis, and all of my colleagues at DTU Fotonik for making my stay at the institute a pleasant and inspiring one.

Finally, I would like to thank my family and friends for their support and encouragements, and especially my wonderful wife for her loving support throughout this journey.

Kgs. Lyngby, Denmark   Anders Rasmussen

October 2013

Publications

Published:

[P1] **A. Rasmussen**, J.Zhang, H. Yu, R. Fu, S. Ruepp, H. Wessing, M. Berger, "*High Capacity Carrier Ethernet Transport Networks*", WSEAS International conference: Recent Advances in Circuits, Systems, Signal and Telecommunications; Jan. 2010.

[P2] **A. Rasmussen**, S. Ruepp, H. Wessing, M. Berger, J.V. Nielsen, H. Hurvig, "*Framed Bit Error Rate Testing for 100G Ethernet Equipment*", HPSR 2010 - 11th International Conference on High Performance Switching and Routing; June 2010.

[P3] **A. Rasmussen**, J.Zhang, H. Yu, R. Fu, S. Ruepp, H. Wessing, M. Berger , "*Towards 100G Carrier Ethernet Transport Networks*", Journal: WSEAS Transactions on Communications, 2010.

[P4] **A. Rasmussen**, S. Ruepp, M. Berger, H. Wessing, *"High-speed Parallel Forward Error Correction for Optical Transport Networks"*, BONE-TIGER2 Summer School, 2010.

[P5] S.Ruepp, H. Wessing, J. Zhang, A. V. Manolova, **A. Rasmussen**, L. Dittmann, M. Berger, "*Evaluating Multicast Resilience in Carrier Ethernet*", WSEAS International conference: Recent Advances in Circuits, Systems, Signal and Telecommunications; Jan. 2010.

[P6] S.Ruepp, H. Wessing, J. Zhang, A. V. Manolova, **A. Rasmussen**, L. Dittmann, M. Berger, "*Evaluating Multicast Resilience in Carrier Ethernet*", Journal: WSEAS Transactions on Communications, Issue 2, Vol. 9, 2010.

[P7] H. Wessing, M. S. Berger, H. M. Gestsson, H. Yu, A. Rasmussen, L. J. Brewka, S. R. Ruepp, "*Evaluation of restoration mechanisms for future services using Carrier Ethernet*", in journal: WSEAS Transactions on Communications , vol: 9, issue: 5, pages: 322-331, 2010.

[P8] H. Wessing, M. S. Berger, H. Yu, A. Rasmussen, L. J. Brewka, S. R. Ruepp, "*Evaluation of Network Failure induced IPTV degradation in Metro Networks*", 4th WSEAS International Conference on Circuits, Systems, Signal and Tele-communications, Cambridge, 2010.

[P9] S. R. Ruepp, H. Wessing, J. Zhang, A. M. Fagertun, **A. Rasmussen,** L. Dittmann, M. S. Berger, *"Providing resilience for carrier ethernet multicast traffic",* 4th WSEAS International Conference on Circuits, Systems, Signal and Telecommunications, Cambridge, 2010.

[P10] Ruepp, S., Rytlig, A., Berger, M., Wessing, H., Manolova, A. V., Yu, H., & Rasmussen, A., *"Evaluation of Speedup and Expansion in Terabit Switch Fabrics"*, In Proceedings of OPNETWORK 2011.

[P11] **A. Rasmussen**, S. Ruepp, M. Berger, H. Wessing, H. Yu, "*SDRAM-based packet buffer model for high speed switches*", OPNETWORK 2011 Conference Proceedings, 2011.

[P12] S. Ruepp, A. Rytlig, M. S. Berger, H. Wessing, A. V. Manolova, H. Yu, **A. Rasmussen**, "*Evaluation of Speedup and Expansion in Terabit Switch Fabrics*", OPNETWORK 2011 Conference Proceedings, 2011.

[P13] H. Hu, J. D. Andersen, **A. Rasmussen**, B. M. Sørensen, K. Dalgaard, M. Galili, M. Pu, K. Yvind, K. J. Larsen, S. Forchhammer, L. K. Oxenløwe, "*Forward error correction supported 150 Gbit/s error-free wavelength conversion based on cross phase modulation in silicon*", Optics Express vol: 21, issue: 3, 2013 .

[P14] H. Hu, D. Kong, E. Palushani, J. D. Andersen, **A. Rasmussen**, B. M. Sørensen, M. Galili, H. C. H. Mulvad, K. J. Larsen, S. Forchhammer, P. Jeppesen, L. K. Oxenløwe, "*1.28 Tbaud Nyquist Signal Trans-mission using Time-Domain Optical Fourier Trans-formation based Receiver"*, Conference on Lasers and Electro-Optics (CLEO 2013), San Jose, California, 2013.

[P15] **A. Rasmussen**, A. Kragelund, M. Berger, H. Wessing, S. Ruepp, "*TCAM-based High Speed Longest Prefix Matching with Fast Incremental Table Updates*", **Winner of "Best Student Paper Award"**, HPSR 2013.

[P16] **A. Rasmussen**, S. Ruepp, M. S. Berger, K. J. Larsen, "*Adaptive Forward Error Correction for Energy Efficient Optical Transport Networks*", HPSR 2013.

Accepted to be published:

[P17] **A. Rasmussen,** H. Yu, S. Ruepp, M. S. Berger, L. Dittmann, "*Efficient Round-Robin Multicast Scheduling for Input-Queued Switches",* IET Networks, 2013.

[P18] **Anders Rasmussen**, Metodi P. Yankov, Michael S. Berger, Knud J. Larsen, Sarah Ruepp, " *Improved Energy Efficiency for Optical Transport Networks by Elastic Forward Error Correction*", Journal of Optical Communications and Networking (JOCN).

Table of Contents

1. Introduction

Global Internet traffic is rising at a remarkable pace. The increasing popularity of bandwidth intensive services such as Internet-based video streaming (Netflix, YouTube, IPTV), video telephony (Skype, FaceTime), and cloud services has provided a demand for very high speed internet connections as well as the applications to utilize bandwidth over long periods of time. Combined with the expanding high speed mobile and land based access networks, this has given rise to an unprecedented increase in internet traffic over the past few years - a trend which is likely to continue for years to come. According to the Cisco Visual Networking Index (CVNI), global internet traffic has increased more than fourfold in the past five years and is expected to increase threefold over the next five years (see Figure 1.1)[1]. When it comes to which kinds of services will dominate the growth in internet traffic, the prediction from CVNI (Figure 1.2), is crystal clear: Video, video, and more video. The slow dialup modems and grainy internet videos are truly a thing of the past. And, where internet based video-on-demand used to be synonymous with internet piracy, there are now plenty of legal content providers, supplying users with high quality content immediately and directly to their high definition televisions or mobile displays.

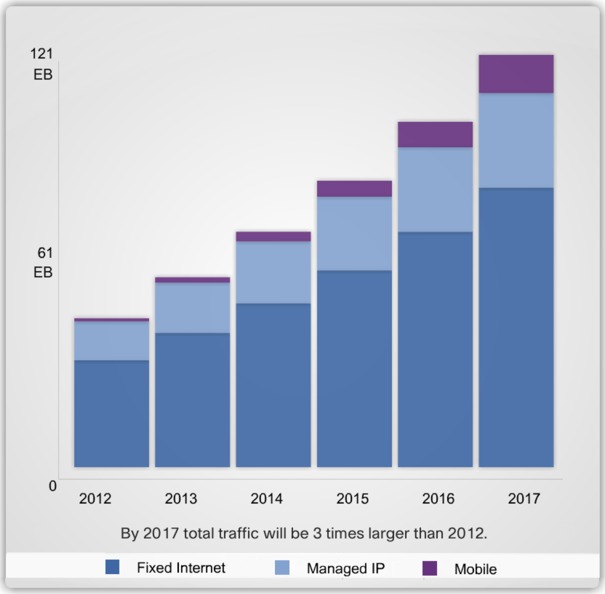

Figure 1.1 – Projected growth in monthly internet traffic 2012-2017 [1].

With this rise in traffic in the access part of the network, there is an increasing demand for high speed connections in the transport networks to handle the aggregated traffic. Instead of simply installing more links using the existing 10G standards (e.g. OTN2 and 10G Ethernet), new standards for 40G and 100G have paved the way for a 4-10 fold increase in the capacities of each logical link. When work presented in this thesis started in May 2009, the 40G and 100G standards were not yet finalized and equipment was still under development [2]. Now, 40G and 100G equipment is commercially available and current standardization and research efforts are focused on extending the 40G/100G architectures to 400G and eventually beyond the 1T boundary [3]–[7].

Increasing the link capacities above 100Gbps impacts the communication systems at multiple levels. The primary focus of this thesis is the challenges which arise at the network node level, when moving to 100G and beyond. Every sub-system, from the switch interconnection fabric and the buffering systems to the packet processors, needs to be redesigned and/or upgraded to support these high speeds. The Danish research project "The Road to 100 Gigabit Ethernet", which ended successfully in May 2012, aimed specifically at meeting these challenges. The aim of the project was to (a) design and implement a 100GE line card with a packet processor and a traffic manager, (b) design and implement a 100GE tester capable of generating and verifying Ethernet framed data at 100Gbps and (c) model and simulate new system architectures and algorithms for 100 Gigabit operation. The three partners in the project, TPACK, Xena Networks and the Technical University of Denmark (DTU) were

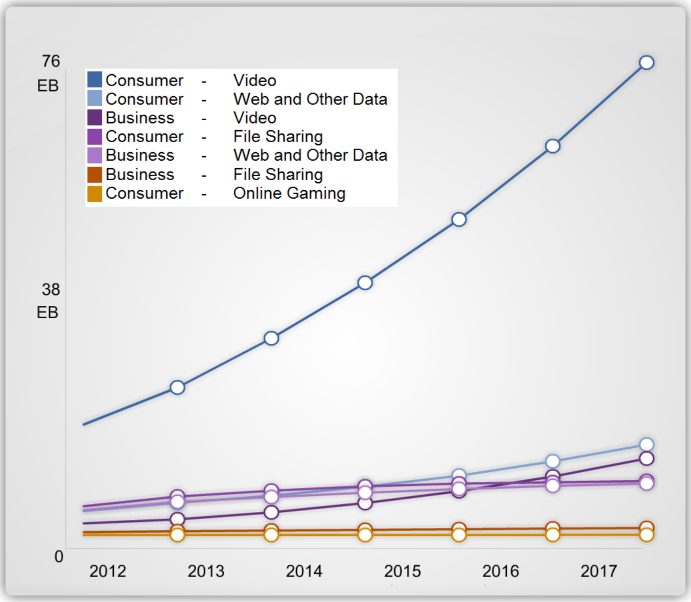

Figure 1.2 – Projected service distribution 2012-2017 (EB/month) [1].

responsible for (a), (b) and (c), respectively. Out of the work done in this project, this thesis focuses specifically on the research within packet processing, in terms of address lookup and packet scheduling, and the development of a 100GE framed bit-error-rate tester (BERT).

At the physical layer, the classical Wave Division Multiplexing (WDM) channel slot size of 50GHz can just barely support 100Gbps with the current modulation formats [8]. Hence in order to reach above this speed, it is either necessary to split up the 400G signal into several serial streams (just like the 100GE standard specifies[9]) or to use larger channels. An alternative is to use more spectrum efficient modulation formats, which in term requires higher signal-to-noise ratios (SNR) and more signal processing to achieve the same bit-error-rate (BER) of the transmitted information. While all aspects of the physical transmission are important, this thesis will limit itself to the forward error correcting codes used at the physical layer to aid in increasing spectral efficiency.

A final aspect of high speed networking, which is important across the layers, is energy efficiency. Energy efficiency is both important from an economical and environmental point of view, but also for pure technical reasons, as high power consumption requires larger power supplies and more effective cooling of the systems. Part of the work leading up to this thesis was done as part of the European CELTIC project “Elastic Optical Networks (EO-Net)” [10]. This project focussed specifically on introducing dynamic power reduction into the transport networks as a mean of trading off throughput for lower power consumption in off-peak periods. Partners in the project included members of academia (DTU, Chalmers University of Technology, Nordunet), equipment / IP-core providers (Alcatel-Lucent Bell Labs, Ekinops, Analogies, C2Tech) and a service provider (France Telecom / Orange labs). Aside from giving a general overview of this work, the thesis describes in detail some of the efforts put into designing and implementing dynamic forward error correction (FEC) for high performance FEC codes.

1.1. Thesis organisation

This section provides an overview of the thesis structure and the content of the individual chapters. Some sections presented here have been written specifically for this thesis, while others are based on published research articles, which have been updated and expanded for the thesis. Of the 16 published and 2 pending journal and conference publications, which were co-authored as part of this Ph. D. study, only the papers most relevant to the research area has been chosen for inclusion in this thesis. The list of papers and their respective sections is shown in Table 1.1.

Chapter 2 gives a general introduction to the challenges of pushing Ethernet towards 100Gbps and beyond, while elevating Ethernet to function as a transport protocol. The chapter also contains a section (2.4) on bit error rate testing on 100G Ethernet connections, attaching the bit error rate tester (BERT) directly to the 100G Media Independent Interface (CGMII). As a prelude to Section 2.4, Section 2.3 provides a brief description of the CGMII and CAUI interfaces and how they differ from the previous 10G standard (XGMII and XAUI). Sections 2.3 and 2.4 are based on papers [P3] and [P2], respectively.

Chapter 3 looks into the issue of performing address lookups at this high speed, specifically on MAC table lookups (Ethernet), and on IP table lookups (Internet Protocol) with special emphasis on a novel TCAM-based scheme for Longest Prefix Matching (LPM). The TCAM section (3.3) is based on paper [P15].

Chapter 4 deals with multicast scheduling and investigates how this can be performed efficiently in high speed, high port count network nodes. This chapter is based on paper [P17].

Section	Papers	Author Number
2.3	[P3]	1st
2.4	[P2]	1st
3.3	[P15]	1st
4	[P17]	1st
5.1	[P18]	1st
5.2	[P4]	1st
6	[P18]	1st

Table 1.1 - Sections of the thesis based on research papers.

Chapter 5 investigates the issue of performing forward error correction (FEC) at high speed. Firstly, the section provides a short introduction to FEC codes, with specific emphasis on the codes used in this project, and introduces the concept of concatenated codes [P18]. Secondly, a case study is described, of a 100Gbps implementation of the Reed-Solomon (255,239) FEC, commonly used in the Optical Transport Network (OTN) protocol, using a Field Programmable Gate Array (FPGA) (based on paper [P4]). Lastly, it is shown how the more complex soft decision Low Density Parity Check (LDPC) FEC can be scaled towards 100Gbps as well.

Chapter 6 examines high speed networks from a power-efficiency point of view with specific emphasis on reducing the power consumption of the FEC circuits (based on paper [P18]).

Chapter 7 concludes the thesis.

2. 100 Gigabit Ethernet and Beyond

In the beginning of the project "The Road to 100 Gigabit Ethernet" (100GE) in 2009, 10 Gigabit links were widely deployed in the transport networks [11]. This is still the case. However, as the internet users' behavioral patterns have moved towards more bandwidth intensive applications, along with the increasing use of cloud services, the demand for extra capacity in the transport networks and data centers have increased correspondingly. The new 100G standards, for OTN (OTU-4) and 100 Gigabit Ethernet (100GE) [9], are capable of fulfilling these bandwidth demands, handling millions of packets per second, and efforts are already being made to develop a new 400G standard[6]. In 2009, when the 100G project started, the underlying hardware technology was just evolved to the point where 100G systems were viable, and even with the recent improvements in chip-technology, memory technology etc., building efficient 100G systems (not to mention 400G systems!) is still no trivial task. The purpose of this chapter is to provide a general overview of the 100GE standard as well as the challenges of designing network equipment, which can operate at 100G and beyond, taking into account recent advances in chip and memory technology.

2.1. The challenge of 100GE and beyond

The main area of research in the 100G project has been the challenges of scaling Ethernet capacity from the 2009 state-of-the-art at 10Gb/s to next generation 100Gb/s. The challenges includes interface adaptation, data and control plane processing, switching, power and printed circuit board requirements. The original goal of 100Gb/s was very ambitious, and required first of all improvements in the low level circuit technology. However, these improvements were far from enough to deliver a factor 10 in performance when moving from 10Gb/s to 100Gb/s, and even less so when looking to future 400G connections.

2.1.1. Basic line card structure

The basic structure of a 100GE line card is depicted in Figure 2.1. The first logical block in on the line card is the transceiver and the Medium Access Control (MAC) block. Since all 100G modules work in full duplex and no auto negotiation is supported, the functionality of the Medium Access Control (MAC) is reduced to Frame Check Sum (FCS) generation and validation (in the TX and RX direction, respectively). The basic functions of the transceivers are described in greater detail in Section 2.3. Once the data has been received and validated, it enters the network processor (NP), which performs classification tasks, such as MAC address and VLAN lookup to determine the forwarding port(s), and Class of Service (CoS) determination. The NP is also responsible for automatic MAC address learning. These functions require lookup tables, which can be accessed very frequently, but which generally do not need to be very large in terms of memory consumption. The issue of address lookup is elaborated further in sub-section 2.1.3. Once classified, the frames move on to the traffic manager (TM). The task of the TM is to determine the QoS of the packets, based on metering and the information provided by the NP, and to shape and prioritize the traffic flows based on this information, as well as other information, such as flow control messages from its peers. Hence, the TM is responsible for the overall scheduling of frame departures into the switch fabric. As such, the TM is also responsible for frame buffering. For the purpose of frame buffering, high throughput and high capacity memory is needed. While access latency is also an issue for the frame buffers, the requirements are not quite as stringent as for the lookup memories. The TM is described in further detail in sub-section 2.1.5. Finally, the frames are relayed to the switch fabric, which interconnects all the line cards in the switch (further details in sub-section 2.1.6).

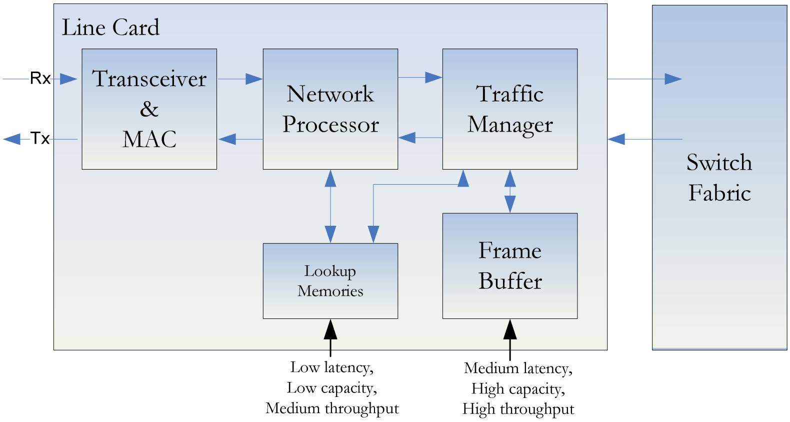

Figure 2.1 – Basic linecard architecture.

2.1.2. Datapath architecture

Since, the basic structure of the Ethernet frames at the MAC level has not changed with the new 100GE standard, an attractive solution would be to simply run the existing 10G designs at 10 times the clock frequency. Unfortunately, despite continuous advances in FPGA performance, increasing the clock frequency is not in itself a viable solution to the problem. Another approach could be to balance the incoming traffic between multiple 10G designs. This allows for reuse of existing logic blocks, running closer to their designed max clock frequency. The trade-off here is increased chip area, higher latency (since each packet is processed slower than 100Gbps), managing resource sharing amongst multiple packet processors (forwarding tables etc.) and a significant challenge in balancing the traffic while avoiding frame reordering. A final approach would be to redesign the system to work with a wider data path i.e. to process more bits in parallel at a lower clock frequency. This poses some challenges to the designer. The authors of the 100GE standard have been kind enough to impose a 8 byte alignment of the Start-of-Frame (SOF) at the CGMII interface[9], which eases the initial frame alignment, even for wide data busses. Unfortunately, the length of the frames can still be any integer number of bytes, which makes e.g. parallel CRC calculation more difficult and poses the potential problem of poor internal bandwidth utilization for packets sizes which are not an integer division of the bus width.

2.1.3. Address lookup

A crucial part of an Ethernet switch is its ability to build and maintain its forwarding table and to perform fast lookups in the table to make forwarding decisions for the individual packets at line speed. To keep up with the worst case traffic patterns of just a single 100GE link, this table must be able to handle up to 298 million lookups per second to handle the combined destination address lookups and source information updates. Furthermore, since switches rarely consist of just a single port, the table structure needs to scale to handle traffic from multiple ports. Popular technology choices for MAC address tables include linked lists, hash tables and Content Addressable Memories (CAMs), depending on the requirements of the system. Due to the importance of address lookup mechanisms, this subject has been given its own chapter in this thesis. Section 3.1 and Sections 3.2-3.3 discuss the issues of address lookups in Ethernet switches and IP routers respectively.

2.1.4. Memory

In a perfect world, all data structures would be kept in high speed on-chip memory. Unfortunately, the large data structures required to perform traffic management, ensure QoS (Quality of Service), perform packet buffering, etc. are too much to keep in on-chip memory alone. Hence, access to high speed external memory is a necessity. This is a substantial challenge when going from 10G to 100G, because the speed of memory does not follow Moore's famous law, stating that the processing power will double around every second year. Especially the larger DRAM-based memories have performance issues when data is stored or accessed in a non-sequential manner (random access), even though there have been significant improvements with respect to the access bandwidth over the years, as seen in Table 2.1. Hence, a combination of different memory types, traffic shaping, and some degree of over-dimensioning of the memory bandwidth is necessary to reach 100Gbps [P11][12]–[15]. When moving on to higher speeds and/or denser systems, this becomes an even greater problem.

For the packet buffers, which require high density memory, i.e. DRAM, current technology is neither fast enough nor compact enough to reach the next goal of 400 Gigabit Ethernet. In order to implement a 400G packet buffer, the effective read/write bandwidth must be at least 800Gbps, assuming perfect bandwidth utilization. Even with state-of-the-art DDR4-2667 memory, this requires five DIMMs, each with 288 I/O pins (see Table 2.1). Hence, a total of 1.440 connections must be routed across the printed circuit board (PCB). The bandwidth bottle neck caused

Technology	BW (Gbps)	Power (W)	mW/GB/s	pj/bit
SDRAM PC133*	8.48	4.96	4664.97	762
DDR-333*	21.28	5.48	2057.06	245
DDRII-667*	42.72	5.18	971.51	139
DDR3-1333*	85.28	5.52	517.63	52
DDR4-2667*	170.72	6.60	309.34	39
GDDR5*	160	2.70	135.00	16.88
LPDDR2-1066 Die	34.08	0.43	100.2/8	12.54
HMC Gen1 512MB Cube	1024	11.08	86.53	10.85

Table 2.1 Comparison of memory technologies[20][120].

by access dependent internal delays inside the DRAM [P11], can to some extent be minimized by clever memory management [13]. Even so, the actual bandwidth requirements are likely to be significantly higher than 800Gbps.

In recent years, the trend within external memory is moving towards memory architectures with high-speed serial interfaces, as opposed to the classical approach of using wide parallel busses[16]–[18]. Using serial interfaces drastically lowers chip I/O overhead and makes it viable to use off-chip packet buffers with enough inter-chip bandwidth to sustain 400G throughput. One such technology is the so called Hybrid Memory Cube (HMC) [19], [20], which has been created by a consortium of developers (Micron, IBM, Altera, Xilinx etc.) and adapters (Napatech, Synopsys, AIRBUS etc.). The HMC is basically a 3D structure as depicted in Figure 2.2, consisting of one logic layer (the controller) and 4-8 layers of DRAM, all interconnected with high speed through-silicon-vias (TSVs). The interface to the memory is through multiple 10G-15G serial links (full duplex). According to the consortium, the resulting memory bandwidth is more than 15x higher than DDR3, using 70% less energy and taking up 90% less space, compared to the common DIMM solution[16]. This is in part a result of the new stacked architecture, the built in controller and a high degree of subdivision of the memory (128-256 banks as opposed to 8 banks in a standard DIMM).

If the promises of the HMC technology hold true, it will go a long way towards satisfying the memory bandwidth requirements and random access performance necessary to make the next jump to 400G. A 2011 demo of the HMC by Micron showed a sustained throughput of just under 1Tbps using 50nm DRAMs and 90nm technology for the logic controller[20]. The newly release specification allows for 32 to 64 links (10G-15G) or up to 128 links (10G). Thus, the maximum aggregated bandwidth supported by this specification (incl. overhead) is 2.56Tbps[19]. Using this technology, the corresponding I/O count for

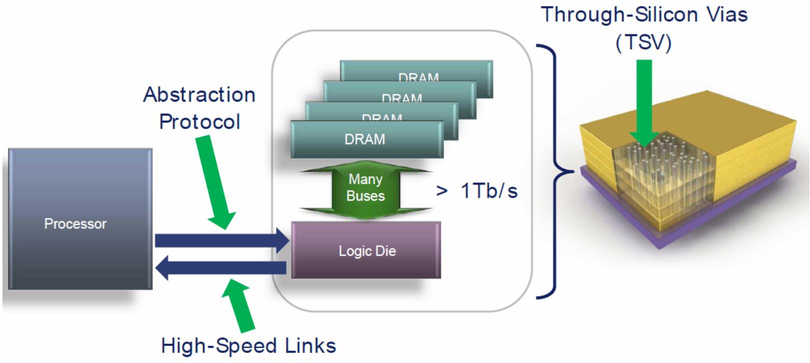

Figure 2.2 – Hybrid Memory Cube (HMC) arcitecture. (HMC height is exaggerated)[20].

supporting 800Gbps of effective read/write bandwidth is 200 pins, assuming 10G differential transceivers and 64B read/writes. The first version of the HMC specification was recently published[19], and according to Micron, the first commercial HMCs will be ready by the end of 2013. Early samples of Altera's first FPGA to support HMC, the Arria 10, will be ready in the beginning of 2014, supporting 1.28Tbps aggregated HMC bandwidth.

As described in subsection 2.1.1, data structures related to MAC address lookup, flow identification, policing, statistics counters and similar functions in the network processor (NP) and traffic manager (TM), do not require a high density memory, but rater memory with excellent random access performance. For this purpose, SRAM based memory is a good choice due to its low and address-independent access latency. One of the fastest SRAM memory types on the market is Quad Data Rate (QDR)-II+ SRAM. QDR uses separate read and write data busses, each of which employs Double Data Rate (DDR) transmission. This eliminates the bus turn-around delay found in shared bus memory types (like DDR SRAM). Currently, QDR memory comes in 9-bit, 18-bit and 36-bit bus width configurations with up to 144Mb of memory. State-of-the-art QDR-II+ XTreme (72Mb) runs at up to 633MHz, corresponding to 316.5 million transfers per second (MT/s) of four words bursts in both the read and the write direction simultaneously [21]. For lower speed applications, Reduced Latency DRAM (RLDRAM) is a viable alternative, providing higher memory density than SRAM with a significantly better random access performance than standard DDR3 RAM.

2.1.5. Traffic management

The task of the traffic manager is to manage the bandwidth of the packet flows based on multiple criteria such as latency, packet loss, bandwidth guarantees, latency and jitter to meet the Quality of Service (QoS) guarantees given to the users. This management includes metering and policing of individual flows, as well as performing traffic shaping, queuing and scheduling are based on this information. The challenge when moving to 100G and beyond is mainly that the processing capacity of the switches in terms of frames per second, as well as the potential number of flows, increases significantly. This poses significant challenges throughout the traffic manager. First of all, the data structures for metering, policing etc., needs to be updated at up to 10 times the frequency of a 10G system. With the high number of flows, these data structures cannot comfortably fit in on-chip memory, and must be stored in high speed external memory such as the QDR-II+ SRAM mentioned in Section 2.1.4. With current state-of-the-art QDR-II+ SRAM, it is possible to perform up to 316.5M read- and write operations per second on 36 to 144 bits (device specific). This allows for more than two 144-bit statistics updates for each frame on a 100G link, assuming minimum sized packets[21]. Given the small I/O count and high performance of QDR memory, this technology is well suited for 100G line cards and appears to be scalable to higher line rates as well.

Another challenge lies in the packet scheduling function. The purpose of the scheduler is to assure that the packets are relayed to the switch fabric in a fair manner, taking the mentioned flow priorities into account. This involves ordering the packets into multiple queues based on metrics such as output port and priority, and arranging the packet departures from the queues by means of a weighted scheduling scheme, e.g. Deficit Weighted Round-Robin or Weighted Fair Queuing [22], [23].

2.1.6. Switch Fabric

Fast processing and efficient queuing is of little help if the underlying switch fabric is not able to keep up with the aggregated line speed. For switches with a relatively low number of ports, the buffered cross-bar architecture yields high switching performance and no blocking. Unfortunately, the number of cross-points as well as the required amount of buffer memory grows proportionally with the square of the number of ports (*$O(N^2)$*), making this architecture impractical for large switch fabrics[24]–[26]. A more scalable approach is employing self-routing switch fabric architectures, such as Clos or Banyan networks. The basic concept of these architectures is to build up the switch fabric as a network of smaller (e.g. two- to four-port) switches. The advantage of this structure is that the total number of cross-points is lower than that of a corresponding full scale cross-bar. The trade-off is some added routing complexity, and in the case of Banyan networks, a small probability of internal blocking[24]. As part of the 100GE project, the Clos-design has been investigated through modeling and simulation[27][P10]. This research revealed only minor performance penalties compared to the crossbar architecture with the benefit of higher scalability. From these results, it seems that self-routing switch fabrics are indeed a viable solution to realize next generation high speed, high port count switches.

Like for the traffic manager, mentioned in 2.1.5, high performance switch fabrics will themselves contain a queuing system and an accompanying scheduler. As opposed to the priority-based TM schedulers, the fabric scheduler has one simple purpose: to assure optimal utilization of the output ports and minimal packet latency under the constraints of the underlying switch fabric and queuing system. While it is always possible to find the optimal scheduling solution given enough time and resources, it is rarely the optimal choice from a practical implementation point of view – especially as the number of output ports increases. Usually, it is more prudent to go for an algorithm, which converges faster (optimally within a fixed runtime) at the cost of lower worst case output utilization and/or higher internal bandwidth requirements. In the case of large scale switches, it is also important to take into account how the calculation complexity scales with the number of ports and how much the scheduling can be segmented to take advantage of hardware parallelism. Section 4 looks further into the switch fabric scheduling problem with special emphasis on efficient scheduling of multicast packets.

2.1.7. Improvements in FPGA technology

Clever design goes a long way towards improving the performance of network devices. However, the 2009 chip technology used to reach 100 gigabit operation cannot support the next step moving towards 400 gigabit connections. Luckily, the chip manufactures have made significant progress in terms of improving the speed, density and power efficiency of their products. In 2009, when the 100GE project started, the then most advanced FPGAs from Altera were selected for implementing the 100G line card and the accompanying tester. The FPGA type used for these two systems was from the Stratix IV family. This FPGA was based on 40nm technology, featuring 530k logic elements (LE) and 24 transceivers running up to 11.3Gbps.

As seen in Table 2.2, there has been a significant development during the last four years. The basic production technology has moved from 40nm to 28nm, with 20nm devices announced for 2014 and a 14nm Stratix 10 device in the pipeline. This has made it possible to produce FPGAs, which are both faster and denser. Hence, the number of available logic blocks and on-chip memory has increased by around a factor of two compared

Year	Device Family	Tech.	LEs*	Transceivers	Memory (Mbits)
Altera Devices					
2009	Stratix IV GT	40nm	530k	24@11.3Gbps	20.3
2011	Stratix V GT	28nm	952k	4@28Gbps 32@12.5Gbps	52
2014	Aria 10 GT	20nm	1150k	96@28Gbps	53
Xilinx Devices					
2009	Virtex-6	40nm	504k	48@6.6Gbps 24@11.3Gbps	32.8
2012	Virtex-7	28nm	778k	16@28Gbps 72@13.1Gbps	68

Table 2.2 - Evolution of FPGA technology[30], [40]

*Xilinx “Logic Cells” converted to the Altera “Logic Elements” based on a 1.125:1 ratio[40].

to the 2009 generations. Likewise, the aggregated transceiver bandwidth has increased by almost a factor of ten in the coming Aria 10 GT device compared to the Stratix IV GT from 2009. With this amount of high speed transceivers, it is possible to run 400Gbps bi-directional traffic through the device (2x16x25Gbps), while supporting 960Gbps of full duplex bandwidth (incl. overhead) to an external HMC memory device (4x16x15Gbps). Hence, from a transceiver bandwidth perspective, the next generation FPGA technology should be able to support 400G operation with external memory queues. A more difficult question is whether the increased speed and logic density will be enough to process information at this speed. According to Altera, they expect the core performance of the next generation chips to scale approximately linearly with the transistor size. This indicates a factor two increase in processing power for the Aria 10 compared to the Stratix IV GT. Along with the factor two increase in the number of logic elements, it will likely be possible to reach 400Gbps processing throughput by careful optimization and parallelization. While initial designs are likely to be FPGA based until the standards are finalized and a suitable architecture has been found, mass production of 400G devices will most likely be done using Application Specific Integrated Circuits (ASICs)[28]. This will drive down the per-unit cost (given enough volume) and allow for a much higher degree of integration with significantly lower power consumption compared to FPGA-based systems.

2.1.8. Conclusions on the challenges of 100 GE and beyond

This subsection has highlighted some of the challenges in reaching 100 gigabit line speed and pushing it further towards the future 400G standards. It is clear that there has been significant improvements in the low level hardware during the past few years since the 100GE project started, especially when it comes to the ASIC and the FPGA technology [29]–[31]. Within memory technology, the classical RAM technologies considered at the beginning of the 100GE project [P1] (QDR SRAM, DDR3-DRAM, etc.) have also improved, but not to the same degree. However, new approaches to DRAM module design which are close to becoming commercially available gives cause for optimism within this field as well. Even with these advances in the underlying technology, from an architectural, algorithmic and digital design point of view, there are still plenty of challenges when moving to 100G and beyond.

2.2. Ethernet as a carrier technology

Ethernet is a very popular and well-known technology, and it is estimated that the majority of all internet traffic either originates or terminates in an Ethernet device [P3]. With the introduction of 10G, 40G and 100G Ethernet interfaces, the standard definitely supports the speeds required to migrate further into the core networks. However, the Ethernet standard has traditionally been targeting enterprise or local area networks (LANs), and thus lacks many of the features required to be considered as a carrier grade technology. In order to address these shortcomings, the IEEE has developed a number of standards, enhancing the functionality of the original Ethernet standard. These include:

- 802.1Q: Virtual LAN [32]
- 802.1ad: Provider Bridging (PB) [33]
- 802.1ah: Provider Backbone Bridging (PBB) [34]
- 802.1Qay: PBB-Traffic Engineering (PBB-TE) [35]
- 802.3ah: Ethernet in the First Mile (with OAM) [36]
- 802.1ag: Connectivity Fault Management (OAM) [37]
- 802.1aq: Shortest Path Bridging [38]

The evolution of the MAC frame format corresponding to these standards is depicted in Figure 2.3. The first issue to be solved is the fact that there is no logical separation of sub-networks within a standard Ethernet LAN. Hence, it is not possible to separate the traffic of different customers. This is also a potential security risk, as all LAN ports are part of the same broadcasting domain. With the introduction of VLANs, it became possible to separate a physical local area network (LAN) into multiple

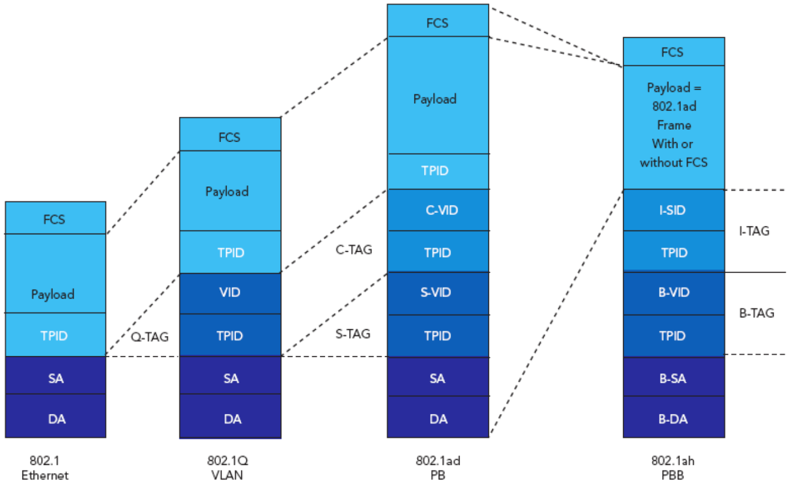

Figure 2.3 – Evolution of the Ethernet header from standard 802.1 to Provider Backbone Briding [P3].

logical networks. However, the limited number of possible VLANs defined in the standard (4,096) was not enough to provide the desired level of separation within the networks. To improve the scalability, so called "Q-in-Q" or "Double Tagging" was introduced with Provider Bridging (PB) in 802.1ad, adding a second tag (S-TAG), which could be used to separate individual customers – leaving the C-TAG for further subdivision, e.g. within a customer's own network. Even so, the size of the S-TAG is still not sufficient for providers with more than 4,096 customers. To meet these and other challenges, the standard has been extended once again to support Provider Backbone Bridging (PBB). PBB takes a layered approach, encapsulating the entire PB frame, including the source and destination MAC addresses, in a larger PBB frame (earning it the nickname "MAC-in-MAC"). The outer PBB frame is only concerned with the transport through the backbone network, and thus uses the MAC addresses of the backbone endpoints (B-SA and B-DA) for forwarding. A B-TAG specifies a flooding domain (virtual networks) within the provider backbone. For logical separation of traffic from different customers, a much larger 24-bit I-SID tag is used, supporting up to 16 million different service instances (customers).

PBB-TE extends on this concept by discarding the flooding/broadcasting mechanisms, as well as the Spanning Tree Protocol (STP) altogether, targeting the protocol towards connection oriented network applications. The B-MAC addresses together with the B-TAG now specify a unique path though the network, using the B-TAG as a means of specifying alternate backup-paths. This adds a measure of resiliency, which is required for carrier class applications such as IPTV, by allowing protection switching in the network and also makes it possible to manually control the exact routing of the traffic. With the recent introduction of Shortest Path Bridging (SPB) to the VLAN standard in 2012, Ethernet now supports shortest path routing between bridge endpoints (e.g. two B-MACs) with multiple active paths. Hence, the backup paths can now be utilized for load sharing to increase performance in meshed networks.

The price for the extra functionality required to use Ethernet in carrier networks is more hardware and software complexity. New functions include flow based queuing and scheduling, advanced routing and forwarding, a more complicated protocol stack and advanced OAM functions, all of which increases the complexity of the network equipment compared to standard Ethernet, which is generally considered a simple and low cost technology. However, by adding more capabilities in terms of customer based Quality of Service differentiation, resiliency, scalability, customer isolation etc., these and other enhancements allow the Ethernet technology to reach into the core networks, instead of being restricted to the edges. There it can serve as a uniform carrier service (much like IP), which can run on native Ethernet links as well as other technologies such as Passive Optical Networks (PONs), WiMAX, SDH/SONET or OTN. Furthermore, it retains its native capabilities such as multicast, easy connectivity (at the customer site), VLANs spanning multiple customer sites, and enables the network operator to increase overall link efficiency in the network by exploiting statistical multiplexing.

2.3. Standardization of 100 Gigabit Ethernet

The 40/100 Gigabit amendment to the Ethernet standard (802.3ba), begun in 2006, was released in its final version in June 2010 [9]. The following section describes some of the basics of the PCS layer and the CGMII interface in accordance with this standard.

2.3.1. 100G Media Independent Interface

Similar to the 10G standard, the 100G standard specifies a Media Independent Interface (MII) for interconnecting the PHY and the MAC. The new 100G MII (CGMII) interface, marked in Figure 2.4, is very similar to the 10G MII (XGMII) interface standard, using the same control sequences to indicate Error (/E), Start-Of-Frame (SOF), End-Of-Frame (EOF) and Idle characters. However, a number of simplifications have been made to the new interface compared to the 10G version. In contrast to the XGMII interface, which is 4 byte aligned, the new CGMII interface is 8 byte aligned, i.e. it uses 8 byte words. This affects functions such as SOF detection, since this character can only appear in the 1st byte position of the 8 byte word. Similarly, Idle sequences after the word containing the EOF character can only be a multiple of 8 bytes. Another restriction is placed on the occurrence of Error characters within the payload. Where XGMII allowed for single error characters everywhere in the binary data, the CGMII interface does not support single error characters in between the data bytes in the frame payload. Hence, if an error is detected in an 8 byte data word, the standard dictates that the

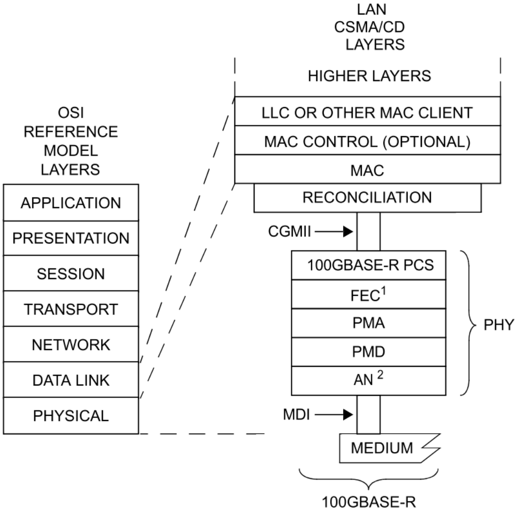

Figure 2.4 - Ethernet Layers (modified from [9]).

entire word should be decoded as error characters. This is due to the 64b/66b encoding format, which unlike the 8b/10b encoding used in XGMII does not allow the decoder to detect single byte or character errors in the payload.

2.3.2. 100G Attachment Unit Interface

As was the case for the XGMII interface, the pin count of the CGMII interface makes it an unviable solution for chip-to-chip or board-to-board communications. Hence, when communicating between devices, such as the MAC and the physical transceiver, a more serialized Attachment Unit Interface (AUI) is used. For the 10G AUI (XAUI), this interface was comprised of four individual physical lanes, each running at 3.125Gbps using 8b/10b encoding. The 100G AUI (CAUI) builds on the same concept, but utilizes 64b/66b rather than 8b/10b encoding to reduce the coding overhead. The CAUI also introduces the concept of *virtual lanes*. Instead of splitting the traffic onto 4 physical lanes, the CAUI distributes the 66b blocks direction between 20 virtual lanes. Splitting the traffic onto 20 lanes makes it possible to adapt the output to several physical bus-width/line-speed combinations by simple multiplexing, such as 1x100G, 4x25G or 10x10G, depending on the physical output device.

In order to keep the different lanes correctly aligned, the XAUI uses a special 8b/10b encoded character, which is sent periodically over all lanes simultaneously. The receiving XAUI uses these alignment characters to verify, that all lanes are properly aligned and to correct the problem, should a misalignment occur. CAUI uses a similar approach, but has extended the alignment markers to fill up an entire 66b codeword on each virtual lane. Aside from a known three byte sequence, which is unique to each individual lane, the alignment markers also include a BIP8 even parity check sum calculated over all the 66b words from and including the last alignment marker. The last four bytes in the alignment marker is simply the inverse of the first four bytes, thus ensuring proper DC balance.

A final difference is the fact that the CAUI uses a scrambler in order to guarantee DC balance and transition density over the 66b words. The first two bits, which determine if the word is a control word or a data word, are kept unscrambled as they are used for 66b word alignment. The alignment markers are likewise kept unscrambled.

2.4. Framed Bit Error Rate Testing for 100GE

As the demand for 40 and 100 Gigabit systems rises, so does the demand for test equipment, which can keep up with this increase in line speed. As with all networking equipment, it is crucial to be able to measure the performance of 100GE systems, during development, deployment and use. A common metric of this performance is the so-called bit-error-rate (BER), i.e. the average ratio between the number of incorrectly received bits and the total transmitted bits count. At the beginning of this Ph. D., BER test equipment for Ethernet systems was only commercially available for speeds up to 40Gb/s [39], [40]. Simply extending these systems to run at 100Gb/s by increasing the clock speed proportionally was not a viable option. This would require extremely fast and expensive hardware and significantly increase the power consumption of such systems. A more viable approach to achieve the required throughput was by means of parallelization, i.e. increasing the number of processed bytes per clock cycle. While this increases the total chip area, parallelization allows for layer 1 bit error rate testing at 100Gbps without pushing the clock speed of such systems beyond reasonable levels.

Another important issue is the fact that the 100GE standard is designed to run over several physical transceivers or SerDes units. The first generations of 100 GE equipment transmit the traffic over several aggregated physical chip interfaces, e.g. 10x10G or 4x25G over a 100 Gigabit Attachment Unit Interface (CAUI) [9]. One approach for BER testing is to simply test the aggregated physical links individually. The 40GE/100GE standard lists this as an optional feature of 40GE/100GE devices and specifies the test parameters that should be used. However, this will not verify the various aggregation mechanisms, such as lane alignment, in the PCS and PMA Ethernet sub-layers, which are handled by the block implementing the CAUI. Nor will it allow the test data to pass through a Device Under Test (DUT) for verification. In order to fully verify the system, on both the tester and the DUT, the bit-error-rate tester (BERT) must be able to verify the aggregated stream as opposed to merely its sub components. Therefore, it must be attached at the 100 Gigabit Media Independent Interface (CGMII), the intra-chip interface between the MAC and the CAUI block (see Section 2.3, Figure 2.4). This poses the additional challenge of framing and deframing the BERT data at line speed to comply with the CGMII standard.

The following sections outline an FPGA implementation of a BERT system, which interfaces to a CGMII. The FPGA technology was chosen over an ASIC implementation due to the lower start up cost, as well their non-static nature, which rendered FPGAs highly suitable for the 100GE field in 2009, where standards were still being finalized. Furthermore, the FPGA platform made it possible to extend the 100GE tester to work on the higher layers (MAC/IP) with limited modifications to the platform. Hence, it was possible to use the 100GE BERT as a stepping-stone towards a more advanced test system based on the final 40GE/100GE standard and continue to mature the product according to evolutions in customer needs. Implementing a 100GE CGMII BERT poses two main design challenges: Firstly, the test data must be validated and generated at the required line speed while keeping the resource utilization down in terms of power demands and logic consumption; and secondly, the test data must be inserted into an Ethernet framing structure at line speed for transport over the CAUI. The following subsections address these challenges, and presents and evaluates the power and logic cost of the FPGA based BERT implementation.

2.4.1. Clock Frequency Versus Buswidth

Before moving on with the actual system design, it is first necessary to consider the trade-off between bus width and internal clock frequency. Since the required serial throughput of the design is very high compared to the achievable internal clock speed in modern FPGAs, it is necessary to introduce a high degree of parallelization. As mentioned in Section 2.1.2, one could simply replicate a slower circuit (e.g. 10Gbps) several times over and multiplex these streams together to form the required 100Gbps of throughput. Unfortunately, this way of parallelizing the system comes with a substantial overhead in terms of buffering, multiplexing, demultiplexing and controlling several aggregated streams, which makes this a far from optimal solution. Instead, this design reaches the required throughput using a single entity with a very wide bus in order to reduce the clock rate to a realistic level. The exact width of the system bus is a compromise between design complexity and clock speed. A larger bus width will decrease the required internal clock frequency of the FPGA, but increases the required FPGA resources and vice versa. Furthermore, the extreme cases of either a very high clock frequency or a very large data bus, heavily complicates the system design, which increases the development cost. In this project, a width/frequency combination of 512bits/195.31MHz has been selected. This lowers the required clock speed to one that can be realistically achieved in current FPGA technology, while keeping design complexity to a minimum. The reason for selecting exactly 512 or 2^9 bits is that bus widths which are a power of 2 are generally easier to work with in digital designs.

2.4.2. Bit Error Rate Testing

The heart of the 100G Bit Error Rate Tester (BERT) is the test pattern generation-and validation mechanisms, which are located on the transmit (TX) side and the receive (RX) side respectively (see Figure 2.5). The test traffic is generated in the transmitter using a Pseudo Random Bit Sequence (PRBS) generator. This component is able to generate a deterministic signal with properties similar to those of a random signal. Since the sequence is pseudo random, the resulting bit stream can be compared to a locally generated sequence at the receiver after transmission. Because the PRBS generator is implemented as a Fibonacci shift register [41], the receiver simply uses the first part of the received bit stream to instantiate its own local generator. Given that these first bits are received correctly, the receiver's PRBS generator is now synchronized with the transmitter, and any bits of the received sequence that deviate from the local sequence are counted as transmission errors. The sequence used in this implementation is the PRBS-31, which is recommended for link verification by the Ethernet 802.3ba standard[9].

Generating a PRBS at 100 Gb/s is a non-trivial task. Using a serial implementation, as depicted in Figure 2.6, which produces only one bit per clock cycle by means of a Linear Feedback Shift Register (LFSR), would require an internal clock frequency of 100GHz, which is not feasible in current FPGAs. A well-known approach for implementing very high speed PRBS generators is to utilize multiple serial PRBS generators in parallel. These are initialized in such a way that the resulting multiplexed output corresponds to the serial sequence [41]. In investigating this approach, a serial generator for a PRBS-31 has been synthesized to run at a clock speed of 908 MHz in the fastest Altera FPGA on the marked in 2010 (EP4S100G2FA0I1) [40], [42]. Even at this very high frequency, a minimum of 111 PRBS generators must be running in parallel to produce 100 Gb/s. The resulting resource usage for the PRBS generators alone, without any multiplexing, retiming and control logic, is listed in Table 2.3 for a design using 128 serial generators running at 781.25MHz. In order to avoid the large resource consumption and high clock speed of such a

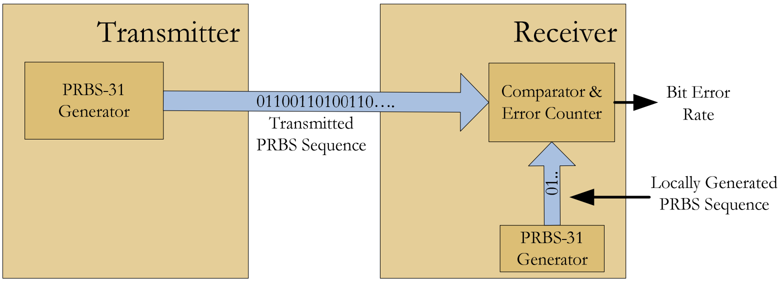

Figure 2.5 - Bit Error Rate Tester (BERT).

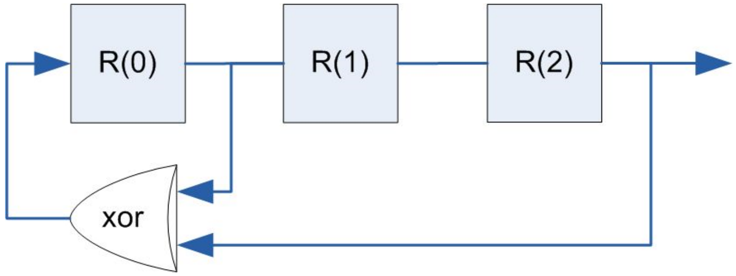

Figure 2.6 - Serial Fibonacci PRBS generator for the polynomial 1 + x + x^3.

system, the parallel PRBS circuit design depicted in Figure 2.7 has been selected instead. This system is based on a serial Fibonacci LFSR [41], but parallelized to generate 512 bits of the PRBS sequence in a single clock cycle. This reduces the clock speed requirement to 195.3125MHz, which is both resource and power efficient and can be relatively easily obtained in commercially available FPGAs, while keeping the system complexity to minimum. The circuit consists of three parts: A 31-bit register, which holds the current state of the PRBS generator, and two combinatorial circuits for calculating the next 512 output bits and the next state of the register based on the current register state. The combinatorial circuits have been designed by pre-calculating the exclusive-OR (XOR) relationships between the current register values and the next 512 outputs of the serial LFSR implementation as well as the next state of the registers corresponding to 512 serial shifts. The calculation of each output bit and each new register value can then be performed in parallel. The resulting parallel generator circuit is able to produce the desired PRBS-31 sequence [9] at 100 Gb/s while keeping both the resource and the power consumption to a minimum.

As seen in Table 2.3, the number of combinational lookup tables (LUTs) and especially the number of registers is dramatically decreased compared

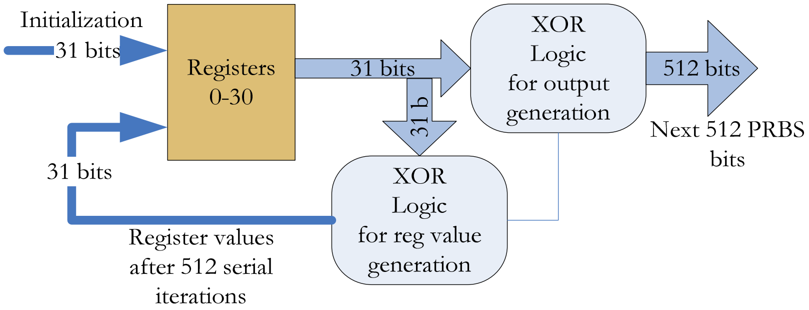

Figure 2.7 - Parallel PRBS generator.

	Serial PRBS generator array (a)	Parallel PRBS generator (b)	Available in the FPGA (c) [42]
Combinational Adaptive LUTs	11.904 (93*128)	564	182.400
Registers	3.968(31*128)	31	182.400
Required clock frequency	781.25 MHz	195.3125 MHz	N/A
Maximum clock frequency	808 MHz	597 MHz	N/A

Table 2.3 - PRBS Generator Synthesis Results

to the approach using serial PRBS generators. The mathematical algorithm used to generate the parallel PRBS generator is described below.

2.4.2.1. Calculating the parallel XOR equations

The parallel equations for the 512bit wide PRBS generator have been auto generated in VHDL syntax using a Matlab program based on the approach described in [43]. The following example uses the simple LFSR depicted in Figure 2.6 with the polynomial x^3+x+1 to illustrate how these equations can be mathematically derived by means of matrix multiplication. Based on the PRBS polynomial, it is possible to set up a transition matrix (T) describing the next state of the registers (i.e. the state after 1 clock cycle) based on their current state:

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad \left| \quad \begin{aligned} R(0) &= R(0) xor R(2) \\ R(1) &= R(0) \\ R(2) &= R(1) \end{aligned} \right.$$

Figure 2.8 - Transition Matrix: T^1.

By multiplying this matrix with itself modulo 2 (T2 mod 2), the equations for the register state after two clock available cycles are obtained:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad \left| \quad \begin{aligned} R(0) &= R(0) xor R(1) xor R(2) \\ R(1) &= R(0) xor R(2) \\ R(2) &= R(0) \end{aligned} \right.$$

Figure 2.9 - Transition Matrix: T^2

Generalizing this approach, it is possible to obtain the equations for the register values after an arbitrary number of clock cycles (q), corresponding to q register shifts. By implementing these equations directly in hardware, the register states can thus be advanced corresponding to 512 serial shifts in a single clock cycle. Since the output of the PRBS generator is always the value of the last register in the LFSR, in this case R(2), the parallel equations corresponding to the next 512 bits in the sequence can be obtained by simply running through all the transition matrixes $\mathbf{T}_q$ from q=1 to q=511 and extracting the 512 equations of the last register. In the example above, the equations for the first output bits would read: Output(0)=R(2), Output(1)=R(1), Output(2)=R(0), Output(3)=R(0) xor R(2), etc. . Using the same algorithm, the PRBS-31 polynomial $x^{31}+ x^{28} +1$ is converted into parallel equations for direct implementation in hardware.

2.4.3. Framed BERT

Since CGMII requires the data to be in an Ethernet frame format, the necessary preamble and inter frame gap (IFG) must be inserted at regular intervals, as depicted in Figure 2.10, to provide framing during transit. At the receiver, the same overhead must be removed to reproduce the original PRBS sequence (Figure 2.11). In order to get a bus width of 512 bits, the native 64bit bus of the CGMII interface is extended eight times, i.e. eight 64bit CGMII words are transmitted over the interface in each clock cycle. The following section will describe the basic operation of the 100Gbit framer/deframer, which communicates with the Ethernet PHY via this interface.

The generated frames consist of a static Start-of-Frame (SOF) and Preamble sequence, a payload field, an End-of-Frame (EOF) character and an idle sequence of variable length. Hence, the PRBS sequence must be broken up into sub-sequences which fit into the payload part of the frames. After transmission, the fragmented sequences must be re-assembled at the other end to reproduce the original stream. The challenging factor is that these functions must be performed on 512 bits (64B) in parallel to reach 100Gb/s.

On the transmission (TX) side of the framed BERT, the framing is performed as depicted in Figure 2.10. In each clock cycle, a 64B vector is delivered from the PRBS generator. The overhead bytes between each frame payload (SOF, idle bytes, etc) are inserted by simply overwriting the generated PRBS bytes in the appropriate positions. The lost PRBS bytes are regained by reinitializing the PRBS generator with the first 31 bits of the overwritten sequence, which will cause the generator to repeat the overwritten bytes as the first part of the next payload. This behavior is inherent to the Fibonacci type LFSR [41], on which the parallel system is based, where the initial register state is always the first 31 bits to exit the generator.

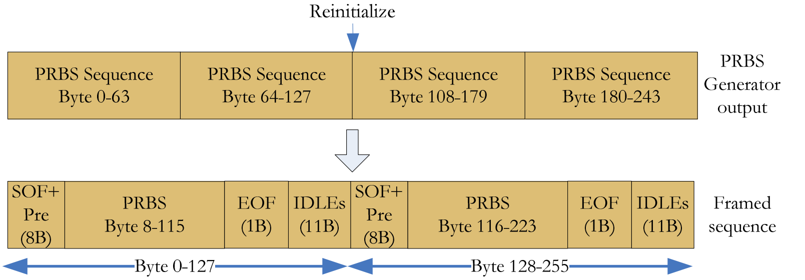

Figure 2.10 - Framing of two adjacent minimum size frames.

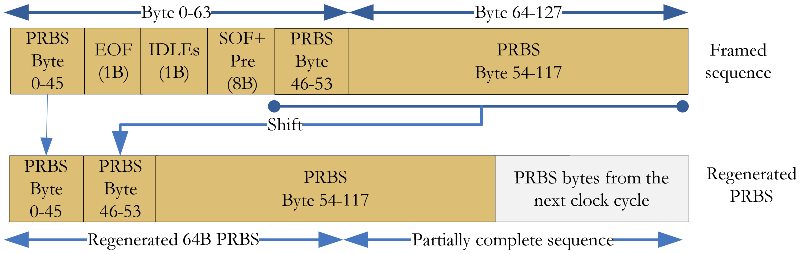

Figure 2.11 - Regenerating the original pseudo random bit sequence.

Framing on the transmission side is simplified by always inserting the overhead bytes in the same positions. On the receiver side of the BERT, one cannot exercise the same control over the exact positioning of the control bytes on the 64B bus. The sequence may be shifted during transmission, and the idle period may be lengthened or shortened to compensate for deviations between the receivers' and the transmitters' local clock signals. As a result, more complex cases exist, like the one depicted in Figure 2.11, where a 64B vector contains PRBS data from two adjacent frames. In order to regenerate the original PRBS sequence, the frames are passed through a system, which deletes the overhead bytes and shifts the fragmented PRBS data together to close the resulting gap. The system takes advantage of the fact that only one sequence of overhead bytes can exist in a 64B vector due to the minimum size of Ethernet frames. This assumption reduces the extraction to a simple shift operation followed by a 2-to-1 multiplexing, a process which can be easily pipelined for higher clock speeds. The resulting PRBS stream is then fed to the bit error rate counter as if it was a serial unframed bit stream.

2.4.4. Synthesis Results

The described system has been synthesized to an Altera Stratix IV GT FPGA (EP4S40G2F40I2) [42]. The entire BERT circuit implementing both the transmitter and the receiver takes up 12,077 Combinational

Adaptive Look-Up Tables (CALUTs) and 10,572 registers with a maximum frequency of 257 MHz. This accounts for the two parallel PRBS generators, the CGMII framing/deframing circuitry and the overall control and glue logic. As anticipated, the deframing function is by far the most critical component in terms of critical path delay and logic utilization. Even so, the proposed design offers a very compact solution, which can be easily pipelined to support even greater throughput or implementation in more modest FPGAs. The system has been successfully and repeatedly verified as part of the first generation Xena 100GE tester, and used to debug other parts of the 100GE system as well as to verify interoperability between the Xena and the TPACK 100GE implementations.

2.4.5. Conclusion on Framed BERT

This section has presented an overview of a novel design for a Bit-Error-Rate-Tester, which is compatible with 100 Gigabit Carrier Ethernet over a 100 Gigabit Media Independent Interface (CGMII). By means of parallelization of the PRBS and framing functions, the BERT has been efficiently synthesized to current FPGA technology with minimal resource and energy requirements. Running bit-error-rate tests over a CGMII interface provides an effective tool for testing both internal PCS/PMA functions as well as providing quality measurements of the logical 100 Gigabit Ethernet links of future transport networks.

2.5. Chapter Summary

This chapter has given an overall introduction to the new 100 gigabit Ethernet standard as well as the challenges of implementing Ethernet equipment with this high line rate. As an example of an actual 100 gigabit Ethernet implementation, the 100G framed BERT is presented, which was a key component in the first proof of concept in the 100GE project. The important subjects of address lookup and scheduling, which have only been mentioned briefly in this chapter, will be elaborated further in chapters 3 and 4.

3. Address Lookup

As mentioned in Section 2.1.3, the ability to find a data entry based on the address information of a datagram is a key feature of most network nodes. These data entries may contain forwarding information, QoS level, statistics etc. This chapter is mainly focused on finding the forwarding information from an address, but the techniques are transferable to the other data structures as well.

When it comes to address lookup, there is a clear distinction between looking for an *exact* match or a *partial* match. *Exact* matches are used in network technologies where the network nodes are expected to know the forwarding port corresponding to each destination address in the network. This is the case for Ethernet switches, which automatically learns and maintains the connectivity information for each peer in the network by monitoring the Ethernet frames as they flow through the switch. A big advantage of exact matching is the relative simplicity of building a binary match/no-match lookup mechanism, compared to selecting the best match amongst several candidates. The disadvantage is that every address must be stored in the forwarding table. Hence, routing based on exact address matching is only viable when the number of possible endpoints is relatively small. *Partial* address matching is used in IP routers. Due to the sheer size of the Internet, IP routers do not have an entry for each reachable IP address in the network, but rather store the network topology in terms of IP sub-network addresses. Along with the common practice of route aggregation [24], this turns IP lookup into a task of finding the entry with the longest matching prefix. This is referred to as Longest Prefix Matching (LPM). As mentioned, this approach makes it possible to store the routing table in a very condensed fashion, which gives great scalability in terms of the number of supportable end nodes. The trade-off is the high complexity of LPM, which makes standard IP routing difficult to scale to high speed core routers. Instead, it is common practice to simplify the lookup operation e.g. by adding a Multiprotocol Label Switching (MPLS) header to IP packets as it enters the core network. The MPLS label specifies the specific route a packet should take across the network and can be forwarded based on exact matching, thus alleviating the scalability issues in core routers. The rest of the chapter is organized into three sections. The first section is focussed on MAC table lookup, i.e. the exact address matching performed in an Ethernet switch. The second and the third give a general introduction to IP address lookup and move on to describe a novel TCAM-based LPM mechanism developed during the course of the Ph. D. project.

3.1. MAC table lookup

The MAC address table in an Ethernet switch is the data structure, which stores the physical port connection corresponding to all the known MAC addresses in the network. The forwarding engine of an Ethernet switch uses this table to determine, which outgoing port(s) an incoming frame should be forwarded to, based on the destination address field in the header. Consequently, the performance of the address table becomes a key factor in the overall forwarding speed of the switch, as well as the switch's ability to forward traffic correctly between a large number of peers. This MAC table is generally formed and maintained dynamically by monitoring the source addresses of the traffic arriving at the input ports (so called MAC learning). Thus, the table must be structured in such a way that new entries are easily inserted as new end nodes are discovered, and stale entries can be removed to support end node mobility and to conserve table capacity. While there are many ways to implement MAC tables, a few of which will be described in sub-section 3.1.1, this chapter will focus primarily on *hash tables* as a mean of storing and retrieving the forwarding information. Specifically, the concept of *chained hashing* [44] and a variation using multiple hash tables [45] will be studied. Finally, based on this analysis, a design proposal to a 100GE MAC table with very low table overflow probability will be presented in 3.1.4.

3.1.1. Background

As described in the introduction to this chapter, the MAC table lookup is performed using exact matching on the MAC addresses. For this purpose, there are several workable data structures from simple RAM based search algorithms to fully parallel hardware search engines.

3.1.1.1. Basic table data structures

The simplest idea for implementing an address table would be to use *direct addressing*, i.e. using the address itself as a memory pointer to the relevant location. For small address spaces, this would be a fast and efficient way to locate an entry, but with the 2^{48} possible MAC addresses, such a table would take up 256TB of memory, assuming 1 byte entries. Hence, it is clear that this approach is not practical for MAC tables. The polar opposite of direct addressing is a *linear memory search*, i.e. the memory is searched one address at the time until the relevant entry is found. While this offers excellent memory efficiency, it has an average search length of half the list size. Hence, it is only a viable solution if the number of peers is expected to be very low. This is rarely the case in Ethernet switches, where the capacity of the MAC table is usually measured in thousands of addresses. A possible alternative to linear search is a *binary search* approach, where the MAC addresses are stored in the table in ascending order of their numerical value [23]. The search starts in the middle of the table and then excludes the top part of the table from the search if the MAC address is smaller than the stored value or the bottom part if it is larger. This process is continued until the search engine arrives at the correct address or finds that the address is unknown. The maximum number of memory accesses in this approach is $log_2(N) + 1$, where N is the number of entries in the table. For 4,096 addresses, this corresponds to 13 accesses. Assuming a memory speed of around 200MT/s (mega-transfers-per-second), this table structure will be able to support destination lookup and MAC learning for five 1Gbps ports receiving minimum sized packets at line speed. Unfortunately, the scheme only works if the table is perfectly sorted. Hence, the removal or insertion of new entries will require a resorting of the entire table, during which time the table will be inoperable. This problem can be mitigated by deferring updates to periods of low activity and/or using two copies of the table – one remains active while the other is being updated. Nonetheless, the strict address order remains a serious drawback of this table structure.

3.1.1.2. Content Addressable Memories

A workable alternative to the RAM-based table structures is *Content Addressable Memories* (CAM)[23]. A CAM is basically a memory array with a large parallel search structure attached to it. This allows the CAM to search for a specific pattern (e.g. a MAC address) in all memory locations simultaneously. The CAM will return the matching address if any, and/or any related information such as the forwarding information for a destination MAC address. Due to their parallel structure, CAMs are capable of extremely high search throughputs, usually one result per clock cycle. The updating procedure is likewise very fast and uncomplicated, since there is no particular structure to the list of entries. Inserting a new address thus reduces to finding a free position (a list of which could be held in a simple FIFO queue) and writing the relevant data to the CAM. Deletions are performed by simply marking the CAM entry as invalid (e.g. by inserting all zeros in the destination address field) and adding the memory address to the list of free position. The lack of structure in the list also means that the CAMs have perfect memory utilization, i.e. a CAM with 8k addresses will be able to store exactly 8k entries, regardless of the dataset. Unfortunately, the extra functionality and high performance does not come without a cost. The parallel search through N memory locations requires N comparator circuits working in concert, the outputs of which must be encoded to form a single match address. This represents a significant overhead in terms of both chip area as well as power consumption compared to regular SRAM memory, and prices have also traditionally been 4-5 times higher for the same amount of CAM memory compared to SRAM prices[23]. This extra overhead also means that the available storage capacities are generally smaller for CAMs compared to SRAMs.

3.1.1.3. Hash tables

In order to reduce the address lookup time without resorting to CAMs, the solution of choice is often a *hash table,* a concept which was first described by Dumey in 1956 [46]. A hash table is basically a compromise between the speed of direct addressing and the memory efficiency of linear search. Since 2^{48} memory locations is obviously unfeasible, these tables use *hash functions* to condense the 48-bit MAC addresses into a smaller pointer of for example 12-bits, which indicates the position of the relevant table entry in a standard RAM module. However, since 2^{48} addresses cannot be mapped uniquely into the 4,096 addresses of a 12-bit address space, collisions will inevitably occur as the number of stored addresses increases. Hence, the entries must include the complete 48-bit MAC address to serve as validation. Furthermore, the issue of several MAC addresses competing for the same table space must be resolved as well.

3.1.2. Related work on hash tables

In research, much emphasis has been put on reducing the lookup time and minimizing the resources usage of hash-based lookup systems by means of Bloom filters, address compression etc. [47]–[49]. Another interesting scheme called Cuckoo Hashing [50] obtains a constant low lookup time at the cost of having a longer and much less predictable address insertion time. However, for address lookup tables in switches, the simpler and more predictable *open addressed chained hashing* scheme [44] is commonly used [51]. Even though several schemes have been proposed to improve the performance of these tables [47]–[49], the Achilles' heel of chained hash tables, i.e. the significant probability of having multiple hash collisions for small hash sizes, still poses a problem, particularly if the table is poorly dimensioned. A 2006 paper by C. Huntley et al. focuses on this issue, particularly on the consequences of hash collisions in Ethernet switches [51]. The paper calculates the theoretical capacity of six commercial Ethernet switches which uses chained hash tables for MAC address lookup and compares the figures with empirical data from experimental tests on the switches. The theoretical and experimental data shows that the actual capacity of the MAC tables in question vary greatly depending on the address distribution of the network. For sequential addresses, all switches meet their specified MAC address capacity, but for random addresses, the tables of these six commercial switches started overflowing at just 6%-39% of their advertised capacities. For sequential MAC addresses, they all achieved 100% of the advertised capacity. The consequence of this is broadcasting of all frames destined to the addresses, which are not stored in the table.

In the following subsections, the hash collision problem will be investigated further by examining the distribution of random addresses over the tables using the popular chained hashing architecture [44]. The results are based on simulation studies using a random generator to produce the MAC addresses on the virtual network. In particular, the distribution of collisions, i.e. how many hash slots have exactly M collisions, as the load factor (the number of stored addresses per total number of hash slots) increases is investigated. This is a key issue, since many chained hash tables have an upper limit on the number of addresses that can be stored at the same hash index. Hence, if the hash table is under-dimensioned compared to the number of addresses it is supposed to store, as in the case in [51], the table will overflow and table entries will be lost. Depending on the physical implementation, long hash chains may also have a negative impact on the lookup speed of the forwarding engine. On the other hand, it is easy to over dimension a hash table in terms of the number of possible hash indexes and/or the number of addresses, which can be stored at each index. This leads to poor memory efficiency, which increases the cost and power consumption of the device without significantly improving the performance. By knowing the statistical distribution of addresses, it is possible to tailor the hash table specifically to the number of addresses it needs to support while balancing performance and cost. Following the statistical analysis, it is shown how a simple parallelization can dramatically increase the hash table throughput without increasing the memory requirements of the table, making chained hash tables a viable solution for the next generation 100 Gigabit Ethernet links.

3.1.1. Chained hash tables

A commonly used way to remedy the problem of hash collisions is to simply allow for two or more MAC addresses to share the same hash key by adding a second layer of memory locations for collisions. This is known as *chained hashing* (CH). Resolving collisions by chaining outperforms simply increasing the size of the hash key from a memory efficiency standpoint, but may require several memory accesses to resolve a single table lookup. A popular approach for pure hardware implementation is to structure the table using *open addressing*, i.e. as a fixed NxM matrix, as depicted in Figure 3.1, where N is the number of hash pointer values (e.g. 4,096 for a 12-bit hash), and M is the number of sub-addresses associated with each pointer. An advantage of this static table structure is that the search can be easily pipelined or parallelized in hardware to improve throughput, thus hiding the performance penalty for doing a linear search through up to M addresses. The price of this fixed memory structure is the large amount of storage space, which is potentially wasted in the upper layers (M>1). Also, if a slot has more than M collided addresses ($M+k$), this must be resolved by either overwriting previously stored addresses or simply not learning the last k addresses. For this reason, it is important that the selected hash function distributes the 48-bit address space fairly evenly over the N hash buckets. Alternatively, an overflow memory such as a small CAM can be used to mitigate the problem, but for any hash function, it will always be possible to find a dataset of X addresses, which will make the table overflow unless $X \leq M$. However, the practical implications of a small number of table overflows in a MAC table are not very severe. If the switch is unable to store a new MAC address, it is by

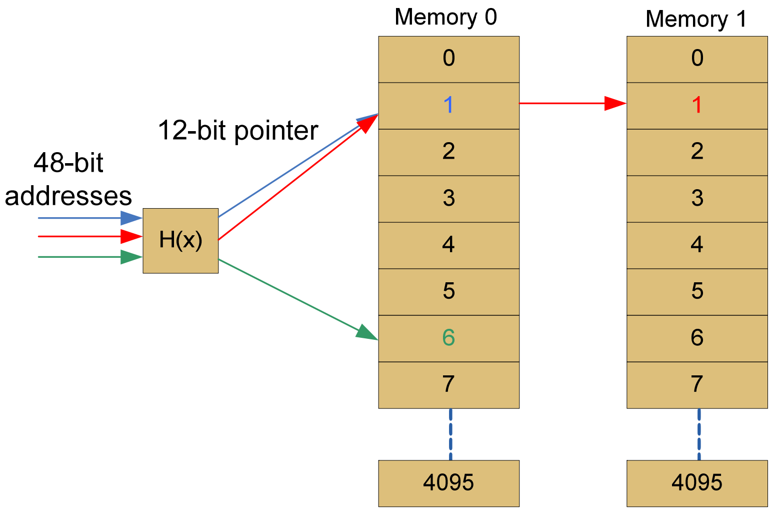

Figure 3.1 – A basic 4096x2 (NxM) chained hash table.

definition unknown, and frames with unknown destination addresses are simply broadcast on all ports (except the port on which it was received). As such, there is no reason to optimize the system for the worst case, but rather the typical case. This can be approximated by means of probability theory or investigated through simulations of the actual hash function and table structure as described in subsections 3.1.2 and 3.1.3.1.

3.1.1.1. Choosing a hash function

When choosing a hash function for the table, there are basically two main criteria:

1. It should distribute the 2^{48} addresses evenly into the N hash buckets to ensure optimal performance and table capacity.
2. It should be efficient in terms of speed and resource consumption.

In literature, there are an abundance of hash functions to choose from, from the very advanced cryptographic hashes, to simple truncation (i.e. using the lower bits as the pointer directly). For a more comprehensive study, the reader is referred to [52][53], which also provides plenty of helpful references. While it may be tempting to go for the very high complexity functions to ensure uniformity, much simpler hash functions will provide adequate performance while reducing both calculation time and hardware complexity. Research has shown, that the common functions used for Cyclic Redundancy Check (CRC), e.g. the CRC-32 used for the Ethernet checksum, are in fact excellent hash functions for the purpose of hash table lookup [49]. These functions can be implemented very efficiently in hardware using a simple Linear Feedback Shift Register implementation and easily parallelized to process a 48-bit MAC address in a single cycle using the same technique described in subsection 2.4.2.1 for the PRBS generator. However, even a simple bitwise XOR of the address bits (folded as ex. 12 by 12 bits or 8 by 8 bits, depending on the required hash size) is in fact an excellent hash function, which provides a perfectly uniform hash distribution [49]. For a 12-bit hash size the equation would be:

$$H_{11-0} = A_{47-36} \oplus A_{35-24} \oplus A_{23-12} \oplus A_{11-0}$$

Aside from the low logic complexity, this hash function also has another advantage; that the table entries only have to store $48 - w$ bits of the MAC address for verification, where w is the width of the table address pointer. This comes from the simple relation between the pointer address and the MAC address bits, which allows the missing w bits to be calculated from the following equation (for w = 12 bits) :

$$A_{47-36} = H_{11-0} \oplus A_{35-24} \oplus A_{23-12} \oplus A_{11-0}$$

Hence, the overall memory footprint of e.g. an NxM table is reduced by $M \cdot w \cdot 2^w$ bits at the cost of an extra XOR circuit.

When using the simpler hash functions such as the folded XOR or truncations, it is important to take the dynamics of the different MAC address bytes into account: the first 3 bytes contain a unicast/multicast indicator bit, which is only relevant to the hash function if the switch supports multicast, and 23-bits which are vendor specific [23]. Hence, in real life networks (and enterprise networks in particular), these bits will most likely be identical for many of the nodes in the network. Therefore, a hash function based primarily on this part of the MAC address will likely perform very poorly. The lower 3 bytes, on the other hand, are usually assigned sequentially by the vendor to each individual MAC unit. Consequently, they can be considered largely random in nature as demonstrated in [49]. In short, it is good practice to include the upper 3 bytes of the MAC address in the hash function, but it is very important to mix them with the lower 3 bytes in such a way, that MACs from the same vendor will still hash uniformly across the hash address space.

3.1.2. Hash Distribution and Performance of Chained Hash Tables

As described in Section 3.1.1, the distribution of the hashed MAC addresses over the N slots in the hash table has a profound impact on the performance of the table, both when it comes to table capacity and table lookup time. This subsection is dedicated to analysing this impact based on simulated hash distributions. The graphs are based on 10,000 simulation runs with different random seeds. For better readability, the confidence intervals have been omitted from these graphs, but for all results, the 95% confidence interval is within ±0.1%. The exact table distribution is highly dependent on the MAC addresses in the network and the hash function, which is used to allocate them into hash slots. However, since the lower 24 bits of the MAC addresses found in real networks are largely random and since hash functions are selected based on their ability to distribute these addresses uniformly, uniform random distribution is assumed in the following analysis. The capacity and lookup time is shown as a function of the table's load factor $\alpha = X/N$, where X is the number of stored addresses and N is the number of hash buckets. This makes the figures generally applicable to tables and data sets of different sizes.

3.1.2.1. Table Capacity

Figure 3.2 shows how the addresses distribute themselves over the different memory levels $M0 - M5$ as a function of the load factor, where $M0$ is the first entry and $M5$ is the sixth entry in the linked search list. As seen from the graph, the MAC table quickly makes use of both the second and the third memory level as the load factor approaches 100%. With a load factor of 200%, over 23% of the addresses need to be stored in level 3 or higher. To illustrate how this can be a problem, let us investigate the scenario with an NxM table, where N = 4096 and M =2 as described in [51]. The maximum capacity of this table given optimal conditions is 8,192 addresses. But since this constitutes a load factor of 200%, less than 73% or approximately 5980 addresses are likely to be stored if there actually are 8,192 peers on the network. The traffic from the remaining 2,212 addresses will therefore have to be broadcasted on all ports. As seen in Figure 3.3, a small amount of flooding is likely to occur for load factors as low as 12.5% for this particular table structure The practical experiments on a device with this table structure in [51] actually showed even worse performance, which may indicate a poor choice of hash function.

When aiming to improve the table capacity (given that the hash function is not the problem), the question is, which of the two table dimensions N and M should be increased to achieve the maximum capacity increase per

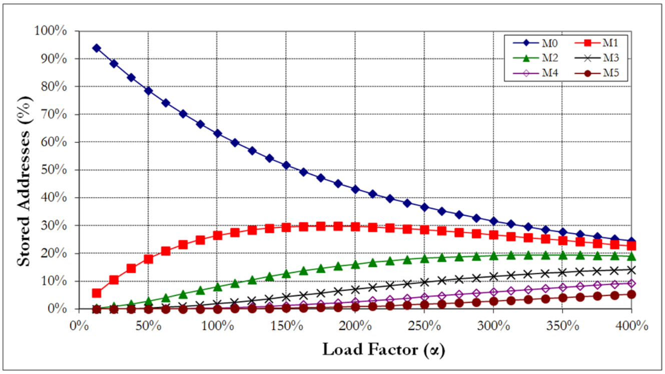

Figure 3.2 – Simulation of the address distionbution as a function of the load factor α, assuming uniform hashing.

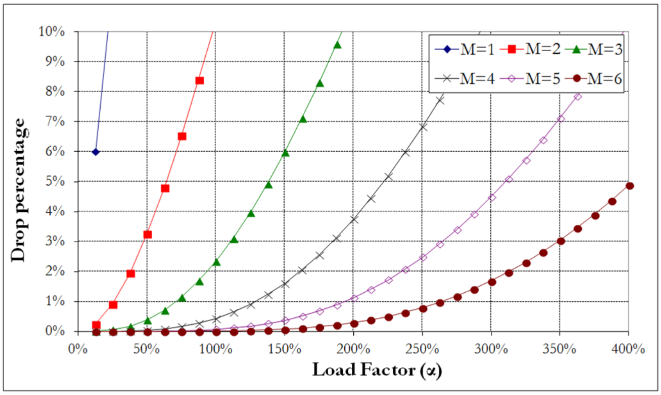

Figure 3.3 – Simulation of the number of unlearned addresses as a function of the load factor and the number if available spaces in each hash bucket (M), assuming uniform hashing.

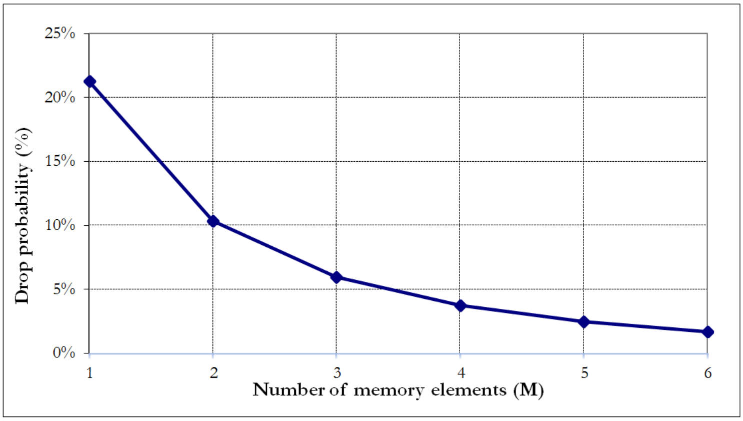

Figure 3.4 –Simulation of the drop probability when storing the same number of MAC addresses in tables with different values of M, but constant total memory size (the NxM product). The load factors for M=1..6 is 50%, 25%, 16.6%, 12.5%, 10% and 8.3%, respectively.

extra bit of memory. Figure 3.4 shows the drop probability for different NxM configurations (with the same total number of entries) when storing the *same* number of MAC addresses. As seen from the graph, the memory efficiency of the table increases for higher values of M, but as described in the next chapter, this may have a negative impact on other performance parameters.

3.1.2.2. Lookup Time

Another important factor when designing a system for looking up MAC addresses is the average lookup time, as it directly affects the frames-per-second (fps) throughput of the switch. Assuming that all the MAC addresses visible to the switch are equally likely to appear in the incoming Ethernet headers, the average lookup time LT_{avg} is calculated as

$$LT_{avg} = \frac{1}{A_t} \sum_{i=0}^{M} A_i \cdot T_i,$$

where A_i is the number of the stored addresses in each memory layer ($0 < i < M$), T_i is the time required for finding an address located at that particular layer (here just measured in number of memory reads) and A_t is the total number of visible addresses. Addresses which cannot be stored in the table due to collisions are added to the $i = M - 1$ contribution, since searching for these addresses (even though they are not found) will require searching through all M sub-tables. The average lookup time determines, how many addresses can be resolved per second for this particular table load, and hence the number of frames per second the MAC table can handle. This again sets an upper bound for the aggregated speed of the ports which share the same table. For a 8-port gigabit switch with a single central MAC table, assuming minimum sized packets and a memory

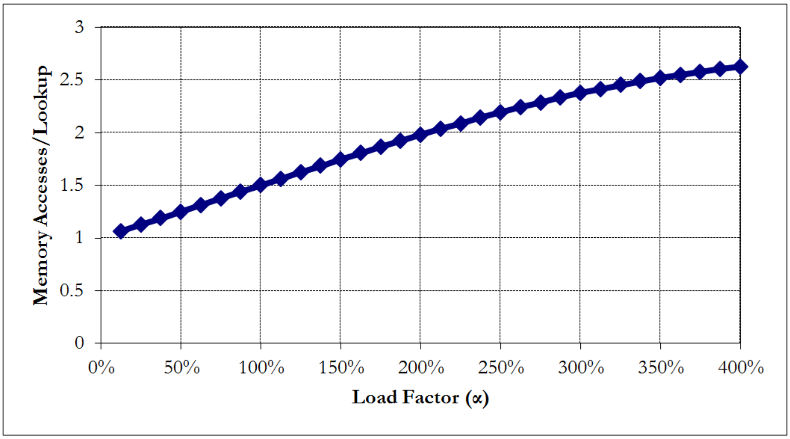

Figure 3.5 – Average lookup time as a function of the load factor for a static NxM hash table.

speed of 125MT/s (mega-transfers-per-second), the average lookup time must be less than

$$\frac{125MT/s}{8 \cdot 2 \cdot 1.49Mfps} = 5.28\,Transfers.$$

Figure 3.5 shows the average lookup time as the load factor increases, based on the hash distribution in Figure 3.2. From the above calculations it is clear that large load factors will not create throughput bottlenecks in small consumer grade switches. In fact, the number of ports could be doubled to 16 in the example above and still meet the lookup time requirements with a load factor of 400%, having an average lookup time of 2.622 memory accesses, assuming homogeneous address distribution. However, this will not scale to enterprise and carrier switches, which will typically support much higher bandwidths and more ports. For these purposes, higher performance designs as the one described in Section 3.1.4 are required.

3.1.3. Multilevel Adaptive Hash Tables

An enhancement to the chained hash table is Multilevel Adaptive Hash Tables (MAHT) (see Figure 3.6). These were first published and patented by Broder and Kalin in 1990 [45], [54]. The MAHT differs from the ordinary chained hash tables (CHT) on three points, each of which improves performance:

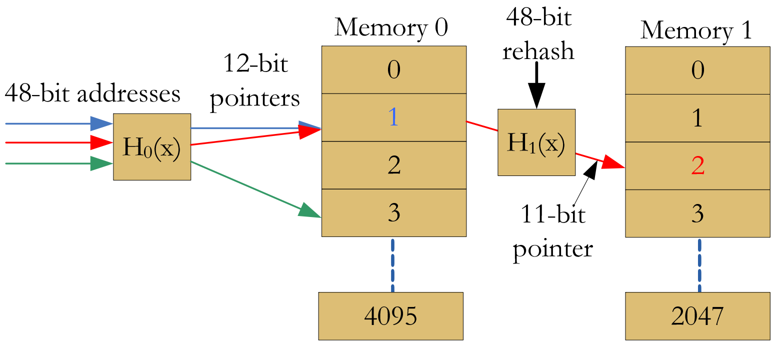

Figure 3.6 – Basic MAHT structure with two tables (memory banks). In event of a collision in table 0, the collided address is rehashed to table 1 using a different hash function. . The tables decrease in size (N) as M increases.

1. Each M sub-tables have their own independent hash function.
2. The tables decrease in size (N) for higher values of M. The authors in [45] propose to use a size of $N_i = N_0/2^{Mi}$, i.e. to reduce the table size by a factor of 2 for each additional table.
3. The hash functions are all dynamic, i.e. they can be changed in case they perform poorly.

Point (1) gives two advantages over CHT. Firstly, it improves memory efficiency in the subsequent $M-1$ sub-tables since they are not limited to store entries in the same positions as M_0. In the CHT case, the memory locations, which are not used in table M_0 will be wasted in the rest of the sub-tables as well. Secondly, the use of multiple hash functions reduces the probability of encountering "bad" datasets which distributes very unevenly over the tables. This is because it is highly improbable (with a good choice of hash functions), that the same dataset will cause bad hash performance in multiple hash functions simultaneously. Point (2) further improves on memory efficiency by leveraging on the fact that the upper sub-tables are usually lightly loaded, and on the results presented in 3.1.2, which show that table capacity can be increased significantly by increasing M at the cost of N. By using progressively smaller sub-tables, the number of these tables can be increased without adding to the total memory footprint. Point (3) is a means of avoiding the case where a particular dataset hashes unevenly in the table (despite the use of multiple hash functions). In this case, the hash functions can be altered and the table rebuilt.

Depending on the chosen platform, the MAHT structure can also have some drawbacks compared to CHT. Firstly, it requires the calculation of multiple hash values, which increases calculation time for software based solutions and chip area for hardware based solutions. Furthermore, where the sub-tables of CHT can be accessed using one single, very wide memory lookup, the MAHT requires either sequential access of each sub-table, or separate memory banks to be accessed in parallel. However, in case of a dedicated hardware solution, these drawbacks are not very severe. With a suitable choice of hash function, like CRC-16 or similar, the hash calculations circuits can be implemented very efficiently. Furthermore, the independent search through M tables makes the scheme well suited for parallelization or pipelining using separate memory banks, preferably on-chip.

3.1.3.1. MAHT vs. CHT Performance

To verify and quantify the expected performance increase when using MAHT instead of CHT, the two types of hash tables have been simulated for different values of N and M. Figure 3.7 shows the drop percentage (percentage of unlearned addresses) as a function of the table load. To make for a fair comparison, the schemes all use around 64k memory locations (MAHT generally use a little less, actual memory usage is listed in the legend). The load is relative to a maximum storage of 64k addresses (100%), corresponding to the maximum theoretical limit for the CHT tables. As seen in the figure, the MAHTs have a significantly lower drop percentages compared to the CHTs, even though they use slightly less memory and have shorter chains. For lower load percentages, the difference is especially significant. At 50% load, the MAHT 32k x 5 table has a drop percentage which is a factor 1,920 and 422 lower than CHT 16k x 4 and CHT 8k x 8, respectively. The scheme also has the advantage of lower drop percentage for smaller values of M, which reduces the number of comparison operations required to perform a lookup. In designs, which store the entire table in a single memory module, the

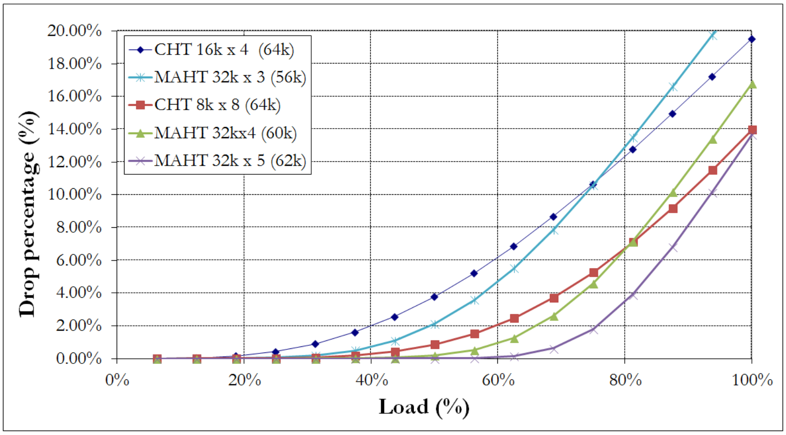

Figure 3.7 – CHT vs. MAHT performance comparison: Simulation of the number of unlearned addresses as a function of the load percentage (100% load = 64k addresses attempted to be stored).

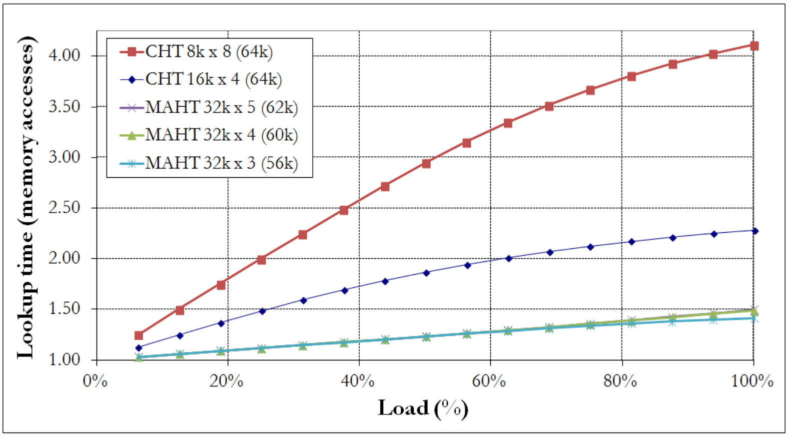

Figure 3.8 – CHT vs. MAHT performance comparison: Simulation of the average number of memory accesses per lookup as a function of the load percentage (100% load = 64k addresses attempted to be stored).

MAHT also has the advantage of lower average lookup times as depicted in Figure 3.8. This is due to the higher utilization of the $M > 0$ memories caused by the rehashing procedure. Hence, more entries are placed in the beginning of the table, resulting in shorter lookup chains on average.

3.1.4. Hash table for 100GE

Due to the very high speed of the interfaces, it is not a viable solution to implement the hash table of a 100GE switch as a central (global) MAC table, but rather to keep distributed local hash tables on the individual line cards. In order to perform MAC address lookup at wire speed on a 100GE connection, the table must be able to perform 148,809,524 destination MAC lookups and source entry updates per second. Since the source updates will need to overwrite the entry information related to the source MAC address after finding the correct address, a regular hash table would require *two* table accesses for each source update operation. However, unless a completely new entry is added, or an existing one needs to be removed, the source update function never needs to read the entry information before updating, aside from validating the MAC address. Hence, the lookup table can be split into two parts as depicted in Figure 3.9: a MAHT MAC table, which is used to find the correct entry address, and a corresponding entry information memory, which holds the relevant information such as port number and aging counters. These two data structures are accessed in a pipelined fashion, i.e. the forwarding engine first finds the correct entry address by performing a MAC search in the MAHT, and then accesses the corresponding address in the entry memory in the following clock cycle. This structure ensures that both the destination lookup and the source update mechanism can have a throughput of *one* operation for each clock cycle. Since no parallel

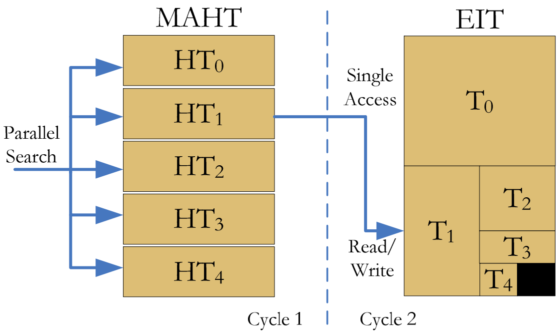

Figure 3.9 – Pipelined memory access. The hash table (left) returns the entry address in cycle 1. The data is read/writte from/to the entry information table (EIT) in cycle 2.

operations are performed on the entry memory, the entry information for all MAHT sub-tables can be contained in only *one* memory bank with the same number of addresses as the combined MAHT sub-tables.

To reduce the lookup time to a minimum, while conserving memory resources, a parallelized version of the MAHT scheme is used with N=64k and M=5 for support of up to 64k MAC addresses. Hence, the sub-table sizes will be 64k, 32k, 16k, 8k and 4k. Assuming good hash functions, the average drop percentage for a fully loaded table will be 0.0019% or 1.24 addresses on average. The individual entries will have a size of 48bits for the MAC address, and 16bits for the entry information (port number, aging counters etc.). To further reduce the memory footprint, the biggest MAHT sub-table will use a simple folded XOR hash function described in 3.1.1.1 and only store 32-bits of the MAC address. The remaining 16-bits will be derived by reverse hashing with the memory address as described. The total size of the table will then be 6.75Mbits, which is reasonable in a modern FPGA or ASIC. The five MAHT sub-tables are each stored in separate banks to allow for parallel access to support 100% throughput independently of the hash distribution. Depending on the available resources and the timing constraints, the memory banks could be run in either single-port mode (requiring a 298MHz clock speed to the memory) or in dual-port mode, where the source- and destination MAC processing can be performed simultaneously (running the memory clock at 149MHz). With this table structure, it is possible to perform MAC table lookup at 100GE line rate with minimum sized packets using just 6.75Mbits of standard SRAM memory. A CAM implementation would require 3Mbits of CAM storage and 1Mbit of SRAM to achieve a similar performance. The design uses 5 or 10 parallel hash circuits (for single- and dual-ported memory access, respectively), which can be implemented efficiently in hardware using simple XOR-logic functions.

3.1.5. Conclusions on MAC table lookup

The MAC table is a core component of an Ethernet switch, and has a significant impact on the achievable packet forwarding rate. This section has investigated hash tables as an efficient and low cost way of implementing this critical address lookup function. However, the structure and dimensions of the hash table needs to be precisely tailored to the specifications of the individual switch in order to keep cost and performance in balance. Therefore, this section has investigated how two different types of open addressed chained hash tables perform for different configurations, in terms of overflow percentage and average number of required memory accesses, as the number of visible addresses increases. Furthermore, it is predicted how often the different parts of the hash table are accessed, which makes it easier to analyze table solutions, where the hash table is spread over several memory modules with varying characteristics (SRAM, SDRAM, CAM). Finally, a design example has been presented for a 100GE distributed lookup table using the Multilevel Adaptive Hash Table (MAHT) scheme, which delivers high performance and low overflow probability with a relatively modest memory footprint of 6.75Mbit per 100GE port. While Content Addressable Memory (CAM) is also a viable solution, offering simpler implementation, higher memory efficiency and dataset independent performance, this is offset by the significant area and power overhead of the parallel search structure. From a throughput perspective, the parallel MAHT can offer just as high performance as the more expensive CAM solution, making it possible to scale this design to 100G operation and beyond.

3.2. IP lookup

Routing tables are the Internet Protocol equivalent of the Ethernet MAC tables. The increase in size of the networks and the lack of address space in IP version 4 (IPv4) requires these routers to maintain large routing tables, possibly with several hundreds of thousands entries [55]. This puts dramatic pressure on implementations for managing these routing tables in the form of insertion of new entries, removal of old entries and searching the tables. The increase in network speed, and thereby the number of packets being received per second, sets further requirements on fast processing. While there are several proposals on how to increase the processing speed of these routers from a software point of view [56], [57], the physical hardware architecture needs to be considered as well in order to ensure scalability and forward compatibility, especially with regards to IP version 6 (IPv6).

IP routers use the routing tables to find the next hop of the packets arriving on the incoming interfaces. The routing table stores information about all the known networks that can be reached in the form of network prefixes. The table is constructed by the exchange of information with connected peers by using a routing protocol such as Open Shortest Path First (OSPF), Border Gateway Protocol (BGP) or similar. Along with each network prefix, the next hop address is stored, which is the interface port or IP address used by the router to forward the packets. The entries in the routing table used for forwarding are stored in a fast local memory in order to achieve as many lookups per second as possible. Traditionally, Dynamic Random Access Memories (DRAMs) have been used due to their low cost and energy consumption. However, the use of other types of memories, such as Static Random Access Memories (SRAMs) and Content Addressable Memories (CAMs), which have much higher performance, have gained more popularity in recent years with continuous increase in chip density and reduction in price [58]–[61].

3.2.1. Common RAM-based data structures for IP lookup

Like for the MAC tables, there are plenty of different data structures for IP forwarding tables. However, as mentioned in the introduction to this chapter, they differ from MAC tables because they have to support Longest Prefix Matching (LPM). One way to perform LPM is using a standard trie structure as depicted in Figure 3.10. It is a binary search trie, which is transversed bit-by-bit based on the destination IP address. Each node corresponding to a network prefix is marked with a flag (an * in the figure), and each time the search arrives at a marked node, the LPM is updated to this value. At the end of the trail corresponding to the particular IP address, the most recent LPM value is used for forwarding.

The issue with the binary trie is that the worst-case number of memory accesses is proportional to the number of address bits. This corresponds to 32 or 128 accesses for IP version 4 (IPv4) or IP version 6 (IPv6), respectively, which is much too slow for gigabit operation.

To improve on this, routing tables often use Patricia tries as depicted in Figure 3.11. These are basically compressed versions of the binary tries, which allows the algorithm to skip over nodes, which do not contain a prefix, and which only has a single child. Still, for large and/or fragmented routing tables, the many memory lookups make them unsuitable for high performance routers[24]. To reduce the number of memory accesses, several approaches have been proposed. One such approach uses a 16-bit prefix table, which points to a position within the tree corresponding to each of the 65.536 possible 16-bit prefixes, hence providing a shortcut towards the correct entry. Unfortunately, this approach does not improve much on the IPv6 performance. Another, more realistic approach is to pipeline the search, using separate memory elements for each level of the trie. This is possible if the trie is small enough to fit in on-chip memory. An issue here is how the memory resources should be split between

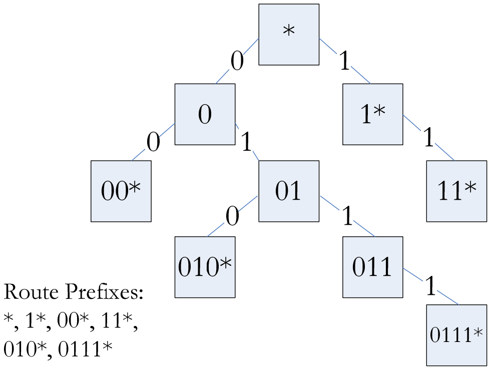

Figure 3.10 – 1-bit trie structure for IP lookup.

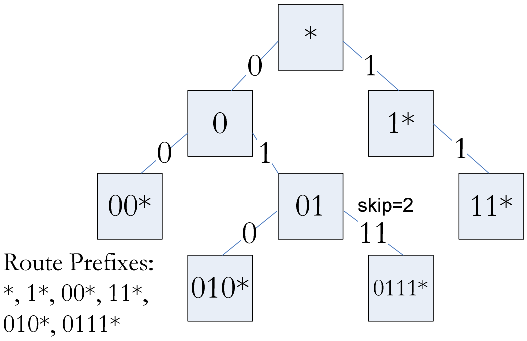

Figure 3.11 – Patricia trie IP lookup.

different levels, as it is not feasible to scale the tree to the $2^{33} - 1$ possible nodes in IPv4, not to mention the $2^{129} - 1$ possible nodes for IPv6.

To improve on the performance of IP lookup mechanisms, the next section investigates Ternary Content Addressable Memories (TCAMs) as a mean of providing throughputs of 100G and beyond in core routers. It further proposes novel enhancements of the standard TCAM design, which optimizes this parallel search engine for Longest Prefix Matching.

3.3. TCAM-based High Speed Longest Prefix Matching

As mentioned in Section 3.1, a Content Addressable Memory (CAM) is basically a large parallel search structure capable of searching through its entire content, generating one search result for each clock cycle. A Ternary CAM has the added feature of allowing wildcards or “don’t cares” in the stored patterns. This is a key feature for the LPM application, which works on partial matches as opposed to the exact matching used in e.g. MAC table lookups. The parallel structure of the TCAM allows it to search through very large data sets in only a few cycles yielding far superior performance compared to the RAM-based search structures. This performance comes at the cost of added chip area and power consumption but if performance is the main priority, TCAMs are a way to get there. Even so, TCAMs do have their drawbacks, as they are not specifically designed with LPM in mind but rather as generic search structures. This section aims to solve these issues by modifying and optimizing the standard TCAM specifically for LPM.

3.3.1. Background

In 1993, McAuley et al. [62] investigated the use of CAMs to increase the speed of the routing table lookups. Since then, a lot of research on how to best utilize CAMs, and especially Ternary Content Addressable Memories (TCAMs), has taken place [63]–[71]. With respect to lookup speed, TCAMs have much higher performance than conventional RAM-based approaches. Even state of the art RAM-based search mechanisms involving hash tables, search tries, bloom filters etc. require several memory accesses for each lookup [72]. When the routing table is stored in a TCAM on the other hand, only one memory access is needed to search through the entire routing table. The drawback however is that the routing table must be arranged in a certain order in the TCAM for correct Longest Prefix Matching. As a consequence, a single route update can trigger a domino effect where large portions of the TCAM need to be updated, during which the forwarding engine will be offline. With core routers receiving many updates per second this is a serious issue, causing latency jitter and packet loss, and a lot of research has therefore focused on how to minimize the impact of having to store the routing table in this certain order [64]–[66], [68], [73]. At the same time, research has focused on how to make efficient memory structures that are most suitable for IP routing tables. Especially the power efficiency of TCAMs is an important topic, as this has traditionally been several times higher than for RAMs of equal size [64].

The required ordering of the routing table comes from the fact that a TCAM will produce multiple results when queried for an IP address, which belongs to multiple network prefixes. In order to distinguish between these results, they must each be given a priority, which is traditionally given implicitly by their location in the TCAM such that entries with lower addresses have higher priority. Unfortunately, this makes it more difficult to update the routing table, as correct ordering must be constantly maintained. The ordering of the forwarding tables implies that the prefixes in the table must be sorted according to the length of the prefixes, such that the longer prefixes appear prior to the shorter prefixes in the TCAM and thus take precedence. This is due to the Longest Prefix Match operation requiring the longer prefix to be used in case of multiple matches. Some methods have been published on how to completely avoid the ordering of the forwarding tables in TCAMs. These ideas typically rely on either separating the forwarding table into different blocks of TCAM based on prefix length [64]–[68], on multiple memory accesses to the same TCAM [62], [74], or on comparison of the lengths of the prefixes [71], [75]. There are advantages and disadvantages to all of these methods. Using several TCAMs may not be very efficient as some prefix lengths will have more entries than others, having to search the same TCAM multiple times reduces the throughput and increases the per packet power consumption while comparing the lengths of the prefixes requires additional logic.

This section describes a novel method for completely avoiding the ordering of prefixes in a TCAM-based routing table by modifying the priority encoder used in the generic TCAM design [76] to perform position independent LPM as part of the match address encoding. The proposed scheme, the Comparator Network LPM (CN-LPM), adds a small area overhead for LPM, but also makes it possible to utilize the TCAM completely since no guard space or table separation is required to support incremental updates. Likewise, the CN-LPM only requires a single TCAM lookup to resolve the next-hop address, which improves both the throughput and the energy efficiency of the forwarding engine compared to schemes relying on multiple memory accesses. As with the generic TCAM design, the area and the power consumption of the proposed scheme scales linearly with the number prefixes ($O(N)$). The rest of the TCAM section is organized as follows. Subsection 3.3.2 describes the typical design of a TCAM-based IP forwarding engine as well as the advantages and drawbacks of such a system. This forms the basis for subsection 3.3.3, which illustrates how the performance can be dramatically improved by modifying the TCAM to better suit the particular application of LPM. The section describes the CN-LPM solution proposed in this thesis and compares it to an existing design by Gamache et al. [75], which uses the same basic concept of enhancing the priority encoder to perform LPM. The Gamache LPM (G-LPM) is used as a benchmark for the CN-LPM design in terms of resource utilization, power efficiency, performance, scalability and ease of implementation. This is described in subsection 3.3.4, which also compares the two circuits to the standard priority encoder commonly used in TCAM designs to perform address encoding and multi-match resolution [76]. Subsection 3.3.5 summarizes the results and concludes the TCAM section.

3.3.2. TCAM-Based IP Forwarding Engine

A TCAM-based IP forwarding engine can be designed as depicted in Figure 3.12. The main components are the TCAM which contains all the network prefixes known to the router, and a Random Access Memory (RAM) which contains the corresponding next hop addresses. Together these two components compose the routing table, which resolves the next hop addresses based on the IP address of incoming IP packets. A lookup operation is performed by scanning through the TCAM for the longest prefix that matches the incoming IP address. The resulting TCAM address is then used to access the RAM, which returns the next hop address for that particular prefix. A TCAM is used as opposed to the simpler CAM because TCAMs support wildcards or *don't-care* values in the stored prefixes. Hence, a 24-bit network prefix for IP version 4 (IPv4) can be stored as a 24-bit binary string followed by 8 don't-care bits. This way, any IP address which matches the 24-bit prefix will cause a match in that particular memory location, regardless of the last 8 bits in the address. However, as mentioned in Section 3.3.1, this property introduces the possibility of multiple matches to a single search of which only the one with the lowest address will be presented to the forwarding engine. This introduces the issue of keeping the table sorted based on prefix length.

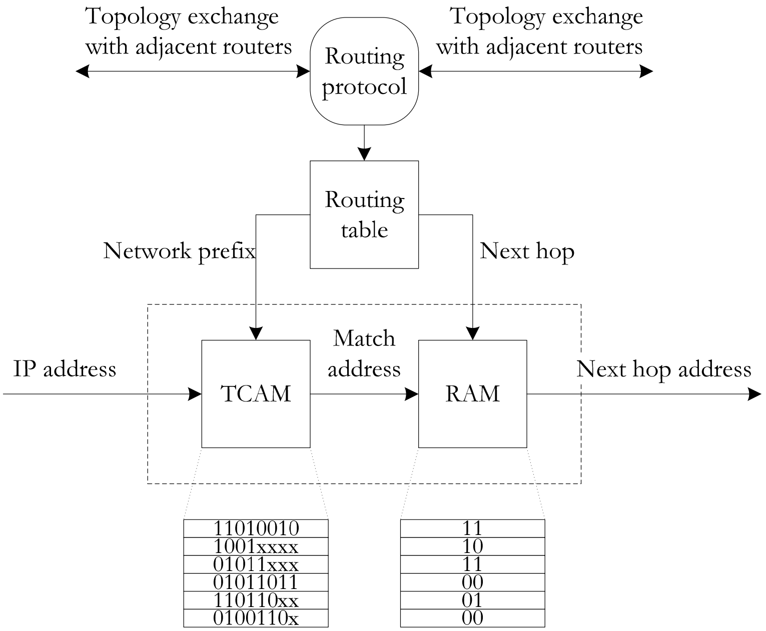

Figure 3.12 - Example of a TCAM-based IP forwarding engine [76].

3.3.3. LPM With Modified TCAM

The generic TCAM can be divided into the matching circuit and the address encoder. The matching circuit consists of N parallel comparators - one for each address in the memory. When a search is initiated, the input data is fed to all N comparators. The output from each comparator is a single match line, which indicates if the input matches the data pattern stored in this particular memory location. Based on these N matching results, the address of the match line with the highest priority is encoded using a simple priority encoder.

The LPM circuits described in this section are based on a modified TCAM as depicted in Figure 3.13, where the basic priority encoder has been replaced by an address encoder, which resolves multiple matches based on the lengths of the stored IP prefixes as opposed to just their relative position in the TCAM. Depending on the IP version, the prefix length can be from 0 to 32 for IPv4 or from 0 to 128 for IPv6. The length 0, which will correspond to *any* address, is the default next hop for packets with prefixes that cannot be found in the routing table. In order to reduce the overhead of storing the prefix lengths, this special case can be implemented as a special exception outside the TCAM by means of a simple comparator and a multiplexer. Hence, in order to support LPM for IPv6, the LPM circuit needs a 7-bit vector associated with each address in

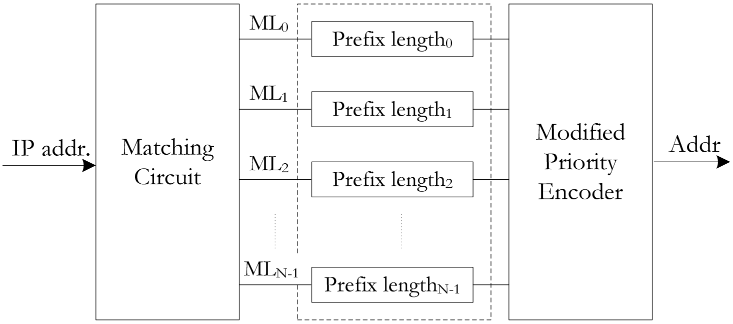

Figure 3.13 - Modified TCAM

the TCAM, corresponding to prefix lengths from 1 to 128. Using this information together with the output of the matching circuit, the address of the match with the longest prefix can be encoded independently of its relative position to similar prefixes in the TCAM and without querying the matching circuit multiple times. This has a significant impact on the throughput and the power consumption of the forwarding engine as well as the memory efficiency of the TCAM compared to the approaches described in subsection 3.3.1. Also, since the relative position of prefixes in the TCAM becomes irrelevant, new prefixes can be easily added or removed at any position in the TCAM without resorting the table or even considering where the information is stored. In the following subsections, the novel CN-LPM architecture proposed in this thesis will be compared to an existing architecture, both of which implements this overall concept in a different way.

3.3.3.1. LPM circuit by Gamache et al.

An existing in-line comparison of prefix lengths in a TCAM structure is proposed by Gamache et al. [75]. In the G-LPM, each TCAM row is followed by special LPM cells which filters out all matches except the one with the longest prefix in a step by step manner as depicted in Figure 3.14. For IPv6, seven LPM cells are used for storing and processing the length of a prefix corresponding to a binary encoding of the values 1-128. The enable signals are asserted one by one for each LPM cell (prefix bit). In each of the seven stages, the LPM circuit will only pass on TCAM matches with an active prefix bit, thus eliminating shorter prefix matches. In case none of the matches has an active prefix bit, all matches are passed on to the next level. After the seven steps corresponding to the binary length of an IPv6 prefix, there will only be a single match within these prefixes. In order to improve the clock speed of the circuit the entire TCAM is divided into 168 blocks of 128 entries each for a total of 21,504 prefixes and the process described above is performed independently within each individual block to produce 168 local winners. To find the longest prefix

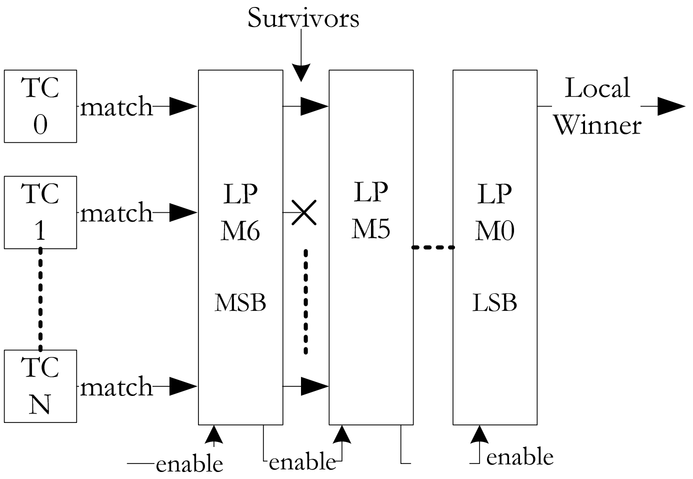

Figure 3.14 - Local LPM circuit for post-processing address mates from *N* TCAM rows.

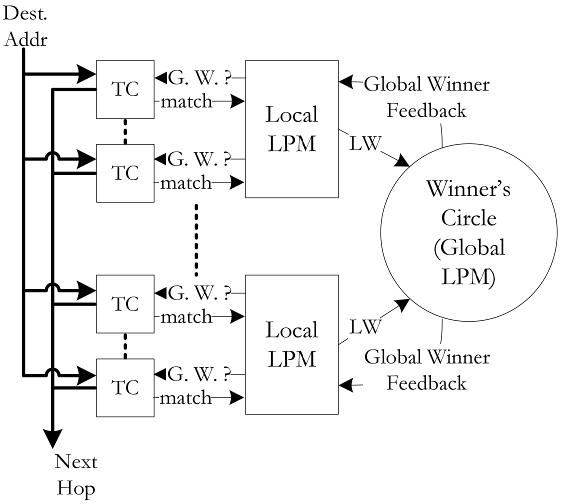

Figure 3.15 - Global LPM circuit with "winner's circle".

match among the 168 blocks, the seven step comparison is repeated in a ”winner's circle” block, as depicted in Figure 3.15, which performs the same operation among the local winners to find the global longest prefix match. When a winner has been found, the single active match line feeding back to the originating row will trigger a readout of the next hop address stored in the corresponding SRAM cells.

3.3.3.2. Comparator Network LPM circuit

The basic concept of the Comparator Network LPM (CN-LPM) depicted in Figure 3.16 is to expand the individual elements of the priority encoder to take the prefix length into account. Like the basic priority encoder, these elements are placed in a tree structure. At the first level, each element takes the match lines and the lengths from two TCAM addresses as an input. Based on this information, the Least Significant Bit (LSB) of the address with the longest prefix which reports a match is encoded and passed on to the next level along with the length of the prefix.

As the active match lines ripple through the levels of the tree, one additional bit is added to the encoded address at each level, until the complete address of the match with highest priority is presented at the output of the last level. Using this address, the corresponding next hop can be read from an SRAM as described in subsection 3.3.2 and depicted in Figure 3.12. The CN-LPM is in many aspects a much simpler design than the G-LPM circuit described in the previous subsection. It features a homogeneous tree structure, which can be easily expanded to support larger or smaller routing tables without changing the basic design. The system only needs $\log_2 N$ levels to support N prefixes and is easily pipelined by inserting registers between the levels at regular intervals.

While the comparator network of the CN-LPM is far superior to the simple location based priority encoder, this functionality comes at the cost of additional complexity in the individual encoding elements. The cost is an additional 7-bit comparator, a 7-bit 2-to-1 multiplexer and a few extra logic gates as illustrated in Figure 3.17 and Figure 3.18. This extra cost must of course be taken into account when evaluating the new design in the following sections.

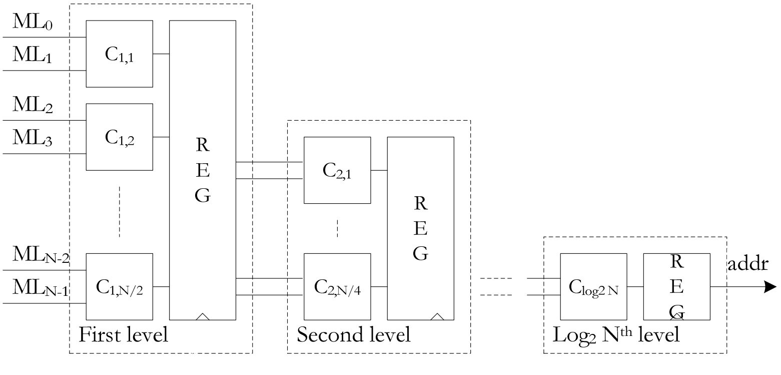

Figure 3.16 - Comparator Network

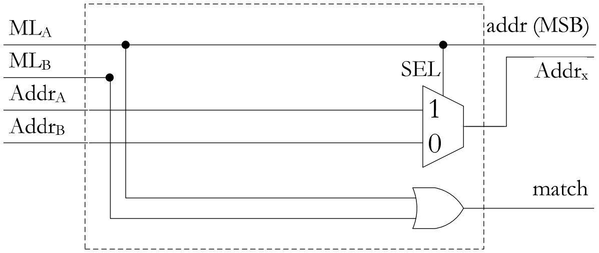

Figure 3.17 - Priority Element

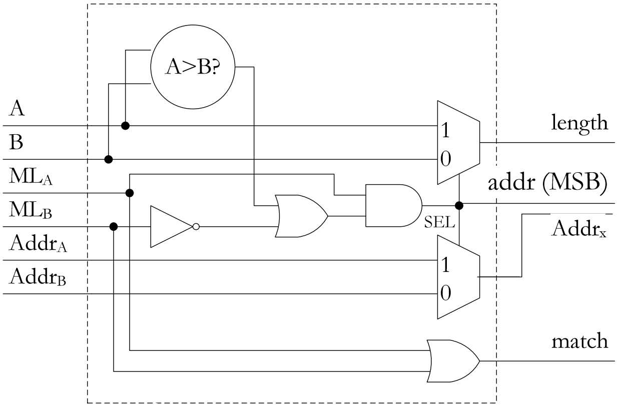

Figure 3.18 - Comparator Element

3.3.4. Resource Utilization and Performance Analysis

In this subsection, the cost of the CN-LPM will be evaluated in terms of area, power consumption and processing speed compared to the G-LPM design proposed by Gamache et al. [75] as well as the priority encoder, which is part of the standard TCAM design. In order to do so, the three systems have been implemented in the hardware description language VHDL and synthesized for a Stratix IV Field Programmable Gate Array (FPGA) from Altera. The three solutions have been compared for 256, 1,024 and 4,096 match lines with the following configurations: All comparators elements are 2x1 (thus comparing only two elements per clock cycle, though still in parallel for all input match lines) in order to achieve the highest possible operation frequency. For the Gamache implementation, the number of inputs per LPM block in both the first and second level is 32 for the results with 256 or 1,024 match lines, and 64 for the results with 4,096 match lines.

3.3.4.1. Logic Cells

The two designs have been compared with a traditional priority encoder in order to determine the ratio between the number of logic elements required for the LPM circuits compared to the priority encoder in an FPGA implementation. As expected, the increased functionality of the LPM circuits does not come for free. In fact, based on 1,024 match lines, the results in Table 3.1 show an increase of 771% and 789% in the number of Adaptive Look-up Tables (ALUTs) for the two LPM implementations, respectively, compared to the priority encoder for a similar number of match lines. The results also show an increase of 334% and 370%, respectively, in the number of registers used. This also includes the registers occupied by the prefix lengths, which are stored in a total of 1,024 × 7 = 7,168 registers. Both the priority encoder and the LPM implementations scale linearly with the number of match lines, so the increase of approximately 770% ALUTs and 350% registers can be expected to remain fixed for larger TCAMs.

	ALUTs	Dedic.Logic Registers
Priority Encoder	2,013	3,061
CN-LPM	15,513	10,223
Difference	**771%**	**334%**
G-LPM	15,887	11,318
Difference	**789%**	**370%**

Table 3.1 - Resource utilization compared to priority encoder (PE) for 1,024 match lines.

While the area increase may seem very significant compared to a standard priority encoder, they are in fact quite modest compared to the size of the TCAM's matching circuitry, which for each prefix needs a total of 128 × 2 = 256 SRAM cells just to store the prefix and the *don't-care* indicators as well as a 128-bit comparator [76]. To ascertain the relation between the resource requirements for the matching circuit and the LPM circuits, the TCAM matching circuit has also been synthesized for FPGA implementation. Compared to the total lookup circuit, the resource requirements of the two multi-match resolution circuits only represent approximately 16% of the total ALUs and 4% of the total registers.

3.3.4.2. Power Consumption

The power measurements were made with 256 match lines, and with a clock period of 2.776 ns (≈ 360 MHz). Using the same clock frequency ensures fair comparison of the power consumption of the different implementations, as the toggle rate of the signals depend on the clock frequency.

Table 3.2 shows the dynamic power consumption of the two different implementations for 256 match lines compared to the priority encoder. The static thermal power dissipation (TPD) has not included since it varies very little for FPGA implementations using the same physical device. The table shows that the CN-LPM has a dynamic TPD which is three times higher than that of the priority encoder. For the G-LPM design from [75], the dynamic TPD is over four times higher. Hence, from a power consumption perspective the CN-LPM is significantly more efficient than the G-LPM implementation by Gamache et al. [75].

	Prio. Enc	CN-LPM	G-LPM
Dynamic TPD	37.57mW	112.43mW	157.43mW
Difference	-	**299%**	**419%**

Table 3.2 - Dynamic thermal power dissipation with 256 match lines.

3.3.4.3. Performance

Since all three circuits have a throughput of one address lookup per clock cycle, the individual performance comes down to the maximum clock frequency at which the circuits can operate. Hence, the maximum clock frequency and the address lookup rate in packets per second (pps) are interchangeable. The lookup rates are specified in Table 3.3 for both 1,024 and 4,096 match lines in order to assess how well the system performance scales with increasing table size. The results in Table 3.3 show that there is an increase of over 110 Mpps between the two LPM implementations in favour of the CN-LPM for 1,024 match lines. For 4,096 match lines, the difference is 65.48 Mpps in favour of the CN-LPM. It can be seen that the G-LPM is generally the slower of the two implementations. A likely cause is the inherent dependence between the individual components in the first levels of the G-LPM, which makes it necessary to implement long and slow interconnection paths between the elements at the same level. Also, the design of the first levels has to be changed depending on the number of match lines which must be supported.

	1,024	4,096
Priority Encoder	734.21Mpps	550.96Mpps
CN-LPM	411.52Mpps	311.24Mpps
Difference	**56.0%**	**56.5%**
G-LPM	300.30Mpps	245.76Mpps
Difference	**40.9%**	**44.6%**

Table 3.3 - Processing rate in mullion packets per second compared to PE for 1,024 and 4,096 match lines.

The CN-LPM on the other hand does not suffer from any of these issues. It is a homogeneous tree structure of similar elements, each of which only depends on the aggregated results from the previous levels. This allows for a simpler design with shorter interconnection paths, which is easily expanded to support larger routing tables. As expected, the priority encoder is by far the fastest of the three designs, given that the contents of the TCAM have already been sorted based on prefix length.

3.3.5. Conclusions on TCAM-based LPM

Section 3.3 has proposed a novel method for performing Longest Prefix Matching in hardware and compared it with the G-LPM design proposal by Gamache et al. [75]. Both designs are built on a TCAM-based search engine which enables search operations in a single clock cycle. These implementations allow for pure hardware based IP lookup, and perform the LPM operation by comparing the lengths of the matching prefixes in the TCAM. The sorting of the prefixes in TCAM-based routing tables is thereby eliminated completely and updates to the TCAM can be performed in a single clock cycle for every update. The CN-LPM scheme presented in this thesis compares the prefix lengths two by two in a comparator network and is able to determine the LPM in $\log_2 N$ clock cycles, where N is the number of match lines in the TCAM. To evaluate this method, a VHDL implementation has been made and synthesised for an FPGA. This has been compared with another VHDL implementation of the G-LPM scheme [75]. For an FPGA application, the proposed CN-LPM method is shown to perform better than the implementation of the G-LPM scheme with less power and area consumption. Compared with the standard priority encoder usually used for multi-match resolving and address encoding in a TCAM, the number of ALUTs and registers required for an FPGA implementation of the comparator network is 773% and 334% higher, respectively. However, when the resource requirements of the matching circuit of the TCAM are taken into account, the ALUT and register requirements of the CN-LPM multi-match resolver remain below 16% and 4% of the total TCAM circuit, respectively. Furthermore, the CN-LPM technique completely eliminates the need for guard space, which would otherwise waste valuable TCAM resources. It has been shown that the CN-LPM is both significantly faster than the G-LPM scheme, with a maximum packet forwarding rate of 411Mpps and 300Mpps, respectively, for 1,024 match lines, and uses approximately 29% less power.

3.4. Chapter Summary

This chapter has shown how the address lookup function, which is a critical component of Ethernet switches, IP routers and other packet based network nodes, can be scaled to 100Gbps operation. Since Ethernet networks do not separate their address space into smaller sub-nets, forwarding is performed using exact matching. As described in Section 3.1, hash tables are very well suited to perform this type of search as a low cost, low power alternative to more advanced search structures such as Content Addressable Memory. It is described how such a hash-based lookup table can be implemented for 100Gbps operation based on the Multilevel Adaptive Hash Table scheme. This table uses only 6.75Mbits of SRAM memory separated into five parallel memory banks to provide 100% throughput with only an expected 0.002% overflow (roughly 1.24 addresses on average) when the table is loaded with 64k MAC addresses. In order to scale to large switches, the table is envisioned as a distributed MAC table, where each 100GE port keeps a local MAC table, which is synchronized with the other port tables periodically.

IP routing tables use the more complicated partial matching scheme called Longest Prefix Matching (LPM) to allow for route aggregation and separation of the IP address space into sub-networks. As a consequence, the concepts used for exact matching in Ethernet are not directly transferable to IP routers. To reach the ambitious goal of over 100Gbps throughput of minimum sized IP packets, the parallel search structure Ternary CAM (TCAM) is proposed for core routers. However, since the standard TCAMs have strict requirements on prefix length ordering in the TCAM memory in order to perform LPM, a modified TCAM design is proposed which removes this requirement altogether. The result is a TCAM with a much simpler and faster updating procedure and 100% memory utilization at the cost of around 16% extra logic compared to the standard TCAM. Trial implementations in an FPGA with 1,024 address entries has shown that this structure is capable of performing 411 million LPM lookups per second, corresponding to a forwarding rate of 411Mpps. If carried over standard Ethernet, this table would be able to handle a sustained line rate of approximately 275Gbps of using minimum sized IP packets.

4. Multicast Switch Fabric Scheduling

This chapter looks into the subject of multicast switch fabric scheduling in input buffered switches. Switch fabric scheduling is the task of coordinating the transfer of packets between input- and output ports across the interconnecting switch fabric in the most efficient way. Efficiency is in this case measured in terms of throughput, delay, fairness, implementation complexity, and scalability. Multicast switch fabric scheduling is an extension for multicast enabled switch fabrics, which aims to utilize the multicast capabilities (i.e. transmitting from one input to multiple outputs simultaneously) as efficiently as possible. This is of increasingly importance as broadcast services, such as cable television, are moving into the IP networks as multicast IP data streams. The chapter first provides an overall introduction to switch fabric scheduling with special emphasis on input queued switches. It then moves on to the primary subject of this chapter, which is efficient scheduling of multicast cells. Specifically, the novel ERRMS multicast scheduling scheme, which has been developed as part of the Ph. D. project, is presented and evaluated through simulation and hardware synthesis.

4.1. Introduction to Switch Fabric Scheduling

The performance of the switch fabric is mainly determined by two parameters: the fabric architecture and the scheduling mechanism. In high performance switch fabrics, packets are often divided into fixed size cells for easier implementation and fairness. It is important that the scheduler is able to schedule enough cells per second to keep up with the line rate while assuring fair and efficient utilization of the fabric bandwidth. The fabric on the other hand must be fast enough to support this cell rate as well as to some extent absorb the performance penalty of non-optimal cell scheduling and filling, and unbalanced traffic distribution. A classical implementation of input queue switches is simple first in, first out (FIFO) buffering, where only the head-of-line (HOL) cell is considered for scheduling. Unfortunately, this strict FIFO architecture suffers from the well known HOL blocking problem, which limits the throughput of the switch to at most 58.6% for uniform traffic[77], regardless of the scheduling. The only way to alleviate this problem is by relaxing the FIFO constraint to allow for more efficient scheduling and/or increasing the internal capacity of the system to compensate for the inefficiencies. Simulation studies have shown that speeding up the system by a factor of two results in 100% throughput, but only for uniform traffic [78].

For a more efficient design, it is necessary to relax the strict FIFO constraints, which opens up for more efficient scheduling of cells across the fabric. One solution is to allow for queue look-ahead, i.e. to base the scheduling decision on the first w cells in each input queue instead of just the HOL cells. The L cells of a queue contend for the next time slot in sequence until an output match has been found or all L cells have been rejected. This approach has only limited impact on the hardware complexity of the fabric and buffering system, and improves the throughput as L increases, but adds to the complexity of the scheduling mechanism, which has to run in an iterative manner. Furthermore, the effectiveness of this approach is reduced under bursty traffic distributions. A different approach is virtual output queuing (VOQ)[24]. Here, each of the N inputs has a separate FIFO queue for each of the N outputs. This completely eliminates the HOL blocking problem, but requires N^2 queues, which can be problematic for very large switch fabrics. Furthermore, the inputs can still only transfer one cell at the time, which can result in idle outputs. Hence, an efficient scheduling algorithm is required to reach optimal performance.

The scheduling algorithm decides which cells should be transferred in each individual cell timeslot. This arbitration is often based on a round-robin principle, where the priorities are rotated amongst the ports in a circular manner. This assures fair and starvation-free scheduling and is relatively simple to implement in practice. To increase the utilization of each cell timeslot, the scheduling algorithm can run iteratively until an acceptable solution has been found. Popular iterative round-robin based scheduling algorithms include iterative round-robin with SLIP (iSLIP) and dual round-robin matching (DRRM)[24]. One issue with the iterative algorithms is their convergence time, which needs to be shorter than the cell time. Due to this time constraint, these algorithms will usually settle for a *maximal matching* solution, where no more input/output pairs can be added without modifying the existing matches, rather than attempt to find the optimal solution (*maximum matching*) through an exhaustive search. Even so, the iterative algorithms are difficult to scale to very high line rates such as 100G without increasing the cell size, and thus sacrificing fabric efficiency. To combat this problem, several schemes have been proposed to speed up the decision rate using pipelining and/or parallelization with promising results[79]–[81].

4.2. Efficient Round Robin Multicast Scheduling

With the basics of general switch fabric scheduling covered by the previous section, the attention is now turned towards multicast scheduling. Multicast is considered an efficient technology for transmitting the same data traffic to a number of network nodes. This is becoming increasingly relevant due to the growing popularity of bandwidth-intensive services such as Internet Protocol television (IPTV), video conferencing and telepresence. In theory, most unicast switches will be able to support multicast traffic with the same general switch architecture and scheduling mechanisms by simply treating the multicast frames as either broadcast frames or multiple unicast frames. However, in order to support multicasting in an efficient manner, both the architecture and the scheduling algorithms of the switch must be designed with multicasting in mind. Much research has been carried out on scheduling multicast traffic in Input Queued (IQ) switches [82]–[90] because of their high scalability. This chapter proposes a new scheduling algorithm, ERRMS, which is designed to reduce the number of times multicast cells need to be retransmitted over the switch fabric in order to reach all output ports, thus reducing the overall load on the fabric. The algorithm is an extension of previous work, the multi-level round robin multicast scheduling (MLRRMS) algorithm described in [87], [89], [90] and has been optimized for more efficient hardware implementation and higher scheduling throughput. Using simulation, the performance of the new algorithm is compared with several other competing multicast algorithms including MLRRMS. In addition to the simulations, the ERRMS as well as the MLRRMS algorithms have been implemented in hardware for multiple port configurations and synthesized for a modern field programmable gate array (FPGA) in order to determine the chip area usage and the scheduling speed of the two algorithms as a function of the port count.

The rest of this chapter is structured as follows. Subsection 4.2.1 elaborates on other works in the related field. Subsection 4.2.2 describes a simplified system model. Subsection 4.2.3 first defines several terms used throughout the algorithm description and then elaborates on and compares the MLRRMS and ERRMS scheduling algorithms. Subsection 0 describes the hardware implementations of the two algorithms. Subsection 4.2.5 shows the performance of the two algorithms in terms of multicast delay, while subsection 4.2.6 shows how the schedulers compare in terms of scheduling speed and resource utilization in an FPGA implementation. Finally, subsection 4.3 concludes on the results from the work on ERRMS and concludes the chapter.

4.2.1. Related Work

The multicast algorithm, TATRA [83], focuses on the IQ architecture, where each input stores its multicast cells in a FIFO queue. After having decided which cells to send, the scheduler leaves a residue of cells to be scheduled in the next cell time. The residue of cells is scheduled based on the departure time, which is the number of cell times before a copy of the cell is served. The TATRA algorithm is strict in fairness and achieves low latency; however, it is high in implementation complexity. In order to reduce the complexity, the weight-based algorithm (WBA) was proposed in [83] as a replacement to TATRA due to its simplicity. The WBA works by allocating weights to input cells according to their age and fan-out before every cell time and choosing the HOL cells with the highest weights to send out. The WBA ensures fairness and has a low implementation complexity but it suffers from the HOL blocking problem [77]. The FIFO-based multicast scheduling (FIFOMS) [91] and the credit-based multicast fair (CMF) [92] scheduling algorithms leverage the virtual output queuing (VOQ) architecture for unicast to schedule multicast traffic in a $N \times N$ switch. N queues are allocated for each input port to schedule multicast cells. Up to N address tokens are generated for each arriving cell, each of which is stored in the queue corresponding to a destination. Each incoming multicast cell is stored in a shared memory and is linked to its address tokens. Based on the scheduling decisions from the scheduling algorithms executed on the address tokens, a multicast cell is sent to the individual destinations and is removed from the memory when all destinations are reached. The FIFOMS and CMF are able to achieve low latency and high throughput, but the bottlenecks of the architecture, especially the token generator and the shared memory, hinder its scalability. The hardware complexity of the address token generator is $O(N)$, since up to N tokens are generated for each arriving cell. In addition, the number of token queues in total is N^2, which can be an obstacle for the switch to scale up in size. The k multicast virtual output queuing (k-MC-VOQ) architecture was proposed in [85]. Each input port maintains k FIFO queues for multicast cells, where $1 < k < 2^N - 1$. The main issues for the k-MC-VOQ architecture are related to the scheduling algorithm and the queuing discipline that associates each multicast flow with a queue. A greedy min-split scheduler (GMSS) [85] was proposed to schedule multicast traffic for the k-MC-VOQ architecture. Each queue is associated with a weight, which is the product of the queue length and the fan-out of the multicast cell at the head of the queue. Queues are examined in descending order of the associated weights. The GMSS iterates with two phases until either all output ports are selected or no more non-empty queues exist at the unselected inputs. With an increase of k, throughput is improved only significantly for small k, i.e. $k \leq N$. Load balancing based on the queue length across multicast queues is required to distribute cells to different queues for performance improvement. It has made the system too complex for efficient implementation.

Another key component and potential bottleneck in the scheduler is the request arbiter and how this is implemented in hardware. The scheme proposed in this thesis is based on round-robin arbitration (RRA), and over the last decade, several hardware designs have been proposed to solve this problem. Shin et. al. [93] has proposed an architecture, where 4×4 round robin arbitration is achieved by dynamically selecting the output of from one of four hardwired fixed priority encoders (FPEs), based on the round-robin (RR) pointer. These 4×4 round-robin priority encoders (RRPEs) are then combined in a hierarchical tree structure to form bigger RRPEs with more ports. While this approach has better performance than dynamically rewiring the in-and outputs using barrel shifters, the multiplexing logic as well as the extra three FPEs results in a significant area overhead. The programmable priority encoder (PPE) proposed in [94] uses only two FPEs in combination with thermometer encoding, a special mask calculated based on the RR pointer. This technique allows for a much more area efficient design, while still providing very good performance characteristics. Jou et al. proposes a more brute force approach to solving the RRA problem [95] by expressing each grant signal in a Sum Of Products form and leaving the rest of the design up to the synthesis tool. Studies in [96], which compares several RRA designs, show that this approach can yield competitive results in terms of timing, but not in terms of area. Furthermore, the long synthesis time is a problem and the authors of [96] were unable to synthesize for 128 ports or above due to the heavy CPU time and memory requirements of the synthesis.

4.2.2. System Model

The switching system is assumed to be an $N \times N$ switch, as described in Figure 4.1. Each input has one FIFO queue to store the incoming multicast cells. The switch fabric is assumed to be bufferless and have intrinsic non-blocking ability, which means that there can always be a connection between an idle input and an idle output. Within one cell time, only one cell can be received by an output. In case that there are multiple cells bound for the same output port, the scheduler must choose which one to serve prior to each cell time. Cells that fail the competition will be stored in the FIFO queue and be scheduled in the following time slots.

It is also assumed that the switch is able to perform fan-out splitting [97], where copies of multicast cells can be delivered to output ports over any number of cell times. The cell will not be removed but remains in the queue, unless all the destinations in its fan-out set are reached. A multicast scheduler makes scheduling decisions prior to each cell time and grants cell transmissions accordingly.

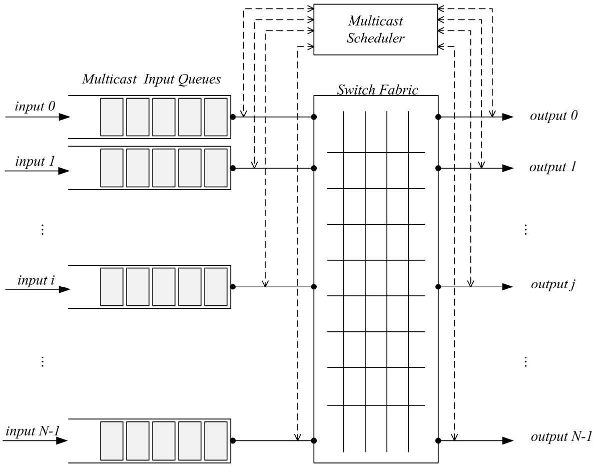

Figure 4.1 - Logical structure of an input-queued multicast cell switch.

4.2.3. Efficient Round-Robin Multicast Scheduling Algorithm

In this section, the motivation of the invention of ERRMS and the detailed algorithm are presented. Several definitions are presented first. Then the MLRRMS on which ERRMS is based will be briefly reviewed and analyzed. Lastly, the new ERRMS algorithm is introduced to tackle the performance bottlenecks of MLRRMS.

4.2.3.1. Definitions

Several terms used in the algorithm are defined as follows:

Definition 1 (Maximum Look-Ahead Depth): The *maximum look-ahead depth*, L, is defined as the maximum number of cells that the scheduler is able to examine behind the HOL cell. Obviously, if the switch is capable of taking an infinite number of cells into consideration for each scheduling process, the multicast HOL blocking problem [89] will be eliminated. However, it is impractical to implement such a system due to the extremely high implementation complexity. To limit the complexity for possible hardware implementation, the maximum look-ahead depth, L, is defined. $L = 0$ indicates that the switch only schedules on the HOL cells in the queues such as WBA [83], else the switch takes up to L cells stored behind the HOL cell per input queue for each scheduling process.

Definition 2 (Cell Position): The *cell position*, p, is defined as the position of a cell in the queue. The cell at the HOL of the queue has $p = 0$.

Definition 3 (Fan-out Vector): A *fan-out vector* is used to indicate the fan-out set carried by a multicast cell in input i at position p, and is denoted as $\boldsymbol{f}^{(i,p)} \triangleq \langle {f_j}^{(i,p)} \rangle, j = 0,1, \ldots, N-1, p = 0,1, .. L, \; {f_j}^{(i,p)} \in \{0,1\}$. ${f_j}^{(i,p)} = 0$ indicates that output j is not in the fan-out set of the cell and ${f_j}^{(i,p)} = 1$ indicates the opposite.

Definition 4 (Traffic Matrix): The *Traffic Matrix* is an $N \times N$ matrix constructed by the scheduler, based on the fan-out vectors of the cells in the position p of each input i, before a cell transmission. It is denoted as $\boldsymbol{T}^{(p)} = \left(T_{i,j}^{(p)}\right)$. Hence, $T_{i,j}^{(p)} = f_j^{(i,p)}, \forall \, i, j, p.\; T_{i,j}^{(p)} = 0, \forall \, j, \; p,$ if input queue i is empty.

Definition 5 (Decision Matrix): The *Decision Matrix* is an $N \times N$ matrix denoted as $\boldsymbol{D}^{(p)} = \left(D_{i,j}^{(p)}\right), D_{i,j}^{(p)} \in \{0,1\}$. This matrix contains the scheduling decisions for each output j, with $D_{i,j}^{(p)} = 1$ indicating that a copy of the cell in input i at position p will be transmitted to output j and $D_{i,j}^{(p)} = 0$ meaning that no copy will be sent to output j.

Definition 6 (Assistant Matrix): The *Assistant Matrix* is an $N \times N$ matrix denoted as $\boldsymbol{A}^{(p)} = \left(A_{i,j}^{(p)}\right), A_{i,j}^{(p)} \in \{0,1\}$. This matrix is used to help generate $\boldsymbol{D}^{(p)}, \; p > 0$

4.2.3.2. The MLRRMS Algorithm

MLRRMS is an iterative algorithm for multicast scheduling, which offers two major improvements over strictly IQ FIFO based schedulers such as WBA. Firstly, the scheduler uses queue look-ahead to increase output port utilization and lower the queuing delay. Hence, any inputs and outputs which are not occupied by the first scheduling round based on the HOL cells $(p = 0)$ will take part in a second scheduling round based on the next cells in the input queues $(p = 1)$. This iterative process will continue for $p = 0 \ldots L$ until all possible outputs have been occupied, all inputs have been served or until the upper iteration limit L has been reached. Secondly, the algorithm uses a special synchronization mechanism which coordinates the individual output schedulers to reduce the number of times individual multicast cells need to transverse the switch fabric. This is done by selecting a *dictator* amongst the output port schedulers in each round. The scheduling decision of the dictator port takes precedence over the decisions of the other ports. Hence, if the dictator grants a request to receive a multicast cell which is destined for other outputs as well, the other outputs will discard their own decisions and follow the dictator. Consequently, the multicast cell will be delivered to all destinations in a single transmission. Simulations on the described system model have shown that this feature significantly improves the multicast performance of the switch in terms of latency and throughput [89]. The dictator is assigned in a round-robin fashion to guarantee fairness. The steps of the MLRRMS algorithm are described below, followed by a specific example.

Initial condition: Before each cell transmission, the cell scheduling position is reset to $p = 0$, i.e. pointing to the HOL cell. All the input and output ports are in unreserved status and are eligible of transmitting and receiving cells.

Step 1) Request: Each unreserved input i submits a request to the unreserved outputs that are contained in the fan-out vector $f^{(i,p)}$ of the cell at the current position pointer p, unless position p is empty. Based on these requests, the traffic matrix $\boldsymbol{T}^{(p)}$ is formed.

Step 2) Dictator Assignment: The dictator for this scheduling round is chosen between the outputs which have received requests in *Step 1*. Dictator assignment uses a round-robin schedule and chooses the first eligible output, starting from the highest priority one, to be the dictator over other outputs. The dictator pointer $a^{(p)}$, which indicates the current highest priority port of the dictator round-robin schedule is incremented (modulo N) to one position beyond the new dictator after the assignment.

Step 3) Masking: Inputs, which have already been served in previous rounds, should not be taken into account and are cleared (masked) from $\boldsymbol{T}^{(p)}$ before scheduling. The resulting Assistant Matrix $\boldsymbol{A}^{(p)}$, containing only requests between eligible inputs and outputs, is passed on to *Step 4*.

Step 4) Individual output scheduling: All unreserved outputs will individually choose amongst the input requests received in *Step 1*. Like the dictator assignment, the decision is performed in a round-robin fashion based on the current RR pointers of the output schedulers. Each queue position (p) for each output has its own RR schedule. Based on the combined scheduling decisions of all of the outputs, the decision matrix $\boldsymbol{D}^{(p)}$ is formed. The RR pointer $d_j^{(p)}$ for the individual outputs, is incremented (modulo N) to one location beyond the selected input, *if and only if*, the individual output decision is not overruled by the dictator in *Step 5*.

Step 5) Sync: The scheduling decision of the dictator, i.e. which input request it has granted, is relayed to the other outputs. Outputs which have received this same input request, but chosen to grant a different one, will change their scheduling decision to match that of the dictator. The $\boldsymbol{D}^{(p)}$ matrix is updated with the new scheduling decisions, thus reserving the granted input/output ports.

Step 6) Look-Ahead: If there are still unreserved input and output ports and if the position pointer has *not* reached its maximum value L, the position pointer increases its value by 1, i.e. $p = p + 1$, and the algorithm returns to *Step 1* to perform the next scheduling round. Otherwise, the scheduling process is completed.

After the scheduling process, each input copies and sends the cells to the corresponding outputs, according to the combined decision matrixes $\boldsymbol{D}^{(p)}$ from each scheduling round. If a cell has reached all the output ports in its fan-out set after transmission, it is removed from the queue. Otherwise, the cell remains in the queue and the newly reached outputs are removed from its fan-out set $f^{(i,p)}$.

A practical example of the MLRRMS scheduling for a 5 x 5 switch with a maximum look-ahead depth of $L = 1$ is depicted in Figure 4.2. In the first iteration, the five inputs $(i = 0..4)$ are requesting to transmit to three of the outputs $(j = 1, 3, 4)$ based on the fan-out vector of their HOL cells (a). From these requests, the Traffic Matrix $\boldsymbol{T}(0)$ is formed (Step 1). Output $j = 4$ is selected as the dictator for this round (Step 2). Since this is the first iteration, no masking is performed (Step 3), i.e. $\boldsymbol{A}^{(0)} = \boldsymbol{T}^{(0)}$ (b). Based on their individual RR schedules, the outputs perform a scheduling decision (Step 4). The dictator grants the request for input 0. Since the $p = 0$ fan-out vector for input 0 also includes outputs 1 and 3, the

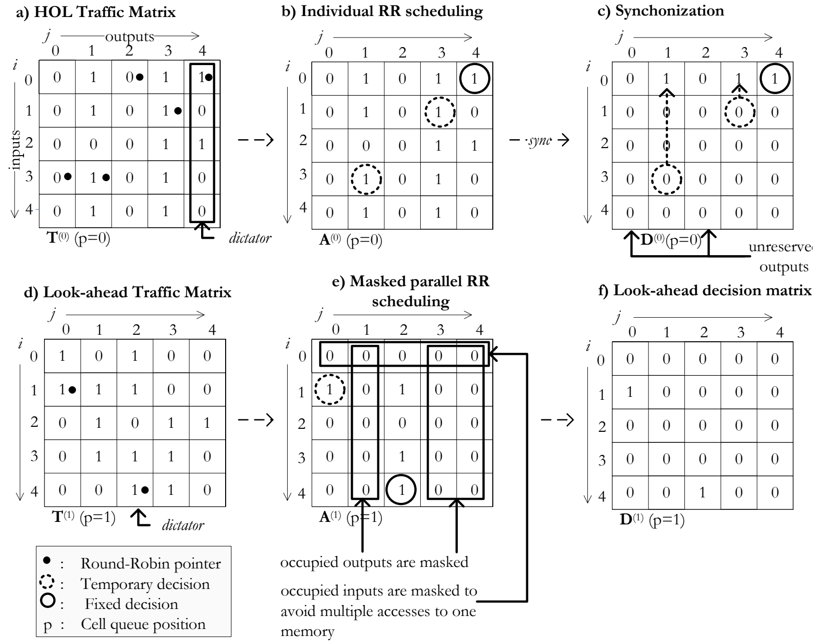

Figure 4.2 - Example of the enhanced FIFO-based round-robin multicast scheduling algorithm (MLRRMS) for a 5 x 5 input-queued switch.

decision of these two outputs are overruled by the dictator in Step 5. Hence, the combined scheduling decision for the first round, $\boldsymbol{D}^{(0)}$, is that input 0 will be allowed to transmit its HOL cell to all outputs in its fan-out vector during the upcoming timeslot (c). After this first iteration, there are still two output ports (0 and 2), which remain unreserved. In order to occupy these inputs, the scheduler will perform a second scheduling round by looking one cell into the input queues (Step 6). The new Traffic Matrix $\boldsymbol{T}^{(1)}$ depicted in Figure 4.2(d) is generated (Step 1) and a new dictator is selected (Step 2). Since input 0 and outputs 1, 3 and 4 are already reserved, the new first matrix $\boldsymbol{T}^{(1)}$ will be masked to only include valid input/output pairs (Step 3). This generates the assistant matrix $\boldsymbol{A}^{(1)}$ (e). Based on this matrix, the two remaining outputs (0 and 2) perform individual RR scheduling with output 2 as the dictator (Step 4). Unlike the first round, the dictator selects an input request, which is not shared by output 0. Hence, output 0 will stick to its individual scheduling decision resulting in the decision matrix $\boldsymbol{D}^{(1)}$ depicted in (f) (Step 5). The final decision of the scheduler is the combination of $\boldsymbol{D}^{(0)}$ and $\boldsymbol{D}^{(1)}$.

4.2.3.3. Efficient Round-Robin Multicast Scheduling

As described, the MLRRMS algorithm searches through each cell position in a sequential manner. This means that the required clock frequency of the scheduling circuit increases linearly with the number of look-ahead iterations for a given cell arrival rate. Furthermore, each iteration relies on results from previous iterations. This inherent data dependency of the algorithm makes physical hardware implementations of the scheduler difficult to scale to higher clock rates, since hardware parallelization or pipelining of the scheduling iterations is not possible. Hence, for hardware implementations, expanding the scheduling past the HOL cell results in a worst case reduction in the schedulings-per-second, which is linearly proportional to L. To fit the algorithm into a high speed switching environment, e.g. 100 Gigabit Ethernet, the ERRMS algorithm tailors the MLRRMS for a faster, more efficient hardware design. The proposed ERRMS algorithm is designed to avoid the data dependencies of MLRRMS to allow for efficient hardware parallelization of the scheduling iterations. Since the resource utilization of the parallel circuit grows as the number of parallel iterations increases, selecting L is a trade-off between scheduling performance and hardware complexity. Through simulations presented in subsection 4.2.5 as well as in related papers [89], [90], it has been verified that the switch is able to provide a maximum improvement-to-complexity ratio on reducing the average multicast delay by allowing the scheduler to examine just one cell beyond the HOL in the input queues, i.e. $L = 1$. Based on this trade-off, a static hardware-oriented algorithm is proposed, which limits its search to $p = 0, 1$. However, the algorithm described here is easily expanded to higher values of L at the cost of more chip area. The new ERRMS algorithm is described below with respect to the original MLRRMS scheme:

Step 1) Submission: Each input queue reads the fan-out information of the first two cells in parallel ($p = 0$ and $p = 1$), and submits this information to the corresponding outputs. $\boldsymbol{T}^{(0)}$ and $\boldsymbol{T}^{(1)}$ are then formed.

Step 2) Dictator Assignment: Different from the MLRRMS, the dictator arbiter of ERRMS only exists for $\boldsymbol{T}^{(0)}$, which means the *sync* function is only carried out on the HOL cells.

Step 3) Decision: Decisions for $p = 0$ and $p = 1$ are performed independently to allow for parallelization. Therefore, masking out previously reserved inputs and outputs is no longer required. To preserve the FIFO priority, cells with the lowest position pointer p always have precedence. Hence, if an output receives fan-out information from both $p = 0$ and $p = 1$ cells, only the $p = 0$ cells will be included in the scheduling. Based on the selected fan-out vectors, each output will individually choose the input request that appears next in its round-robin schedule, starting from the highest priority element. To reduce hardware complexity, the round-robin pointer d is now shared by both levels ($p = 0, 1$).

Step 4) Sync: Since the dictator only exists for $\boldsymbol{T}^{(0)}$, i.e. the HOL cells, the outputs that only receive fan-out information for $p = 1$ cells are exempt from the synchronization process. Otherwise, synchronization is performed in the same way as described in subsection 4.2.3.2.

Unlike the MLRRMS which iterates through the FIFO queues until all eligible inputs and/or outputs are reserved or until the maximum look-ahead depth is reached, the ERRMS executes the four steps described above in a single iteration. Similar to the MLRRMS, the inputs will transmit cells to the outputs after the scheduling process, according to the combined decision matrix $\boldsymbol{D}$ for $p = 0$ and $p = 1$, after which the fan-out vectors will be updated. The independent scheduling of the two queue positions means that an input can potentially receive two grants with different p values, which means that the input is scheduled to send two cells during one cell time, one from $p = 0$ and one from $p = 1$. Thus, the bandwidth of the buffer memory and the switch fabric must be able to support this to implement the ERRMS algorithm.

4.2.4. Hardware Implementation

To determine the relative resource utilization and scheduling speed of the MLRRMS and ERRMS, the two algorithms have been implemented in hardware for an FPGA. Both circuits have been designed for a look-ahead depth of $L = 1$, but are easily expanded to support higher values of L. This section describes the hardware designs for the MLRRMS and the ERRMS. Furthermore, the design of the internal round-robin scheduler is also described since it is a critical common component of the two schedulers, both in terms of hardware complexity and scheduling speed.

4.2.4.1. The MLRRMS implementation

The MLRRMS depicted in Figure 4.3 is an iterative design, which alternates between scheduling cells from $p = 0$ and $p = 1$ with individual round-robin and dictator pointers for the two positions. In the first round ($p = 0$), the mask $\boldsymbol{M}$ is empty and the Traffic Matrix $\boldsymbol{T}^{(0)}$ is therefore passed on unaltered to the round robin priority encoders (RRPE), $\boldsymbol{A}^{(0)} = \boldsymbol{T}^{(0)}$. From the $\boldsymbol{A}^{(0)}$ matrix, the RRPE generates a preliminary decision matrix $\boldsymbol{D}'^{(0)}$ corresponding to the individual round-robin scheduling decisions of the outputs. Based on the decision of the current dictator, the *Sync* block modifies this matrix as described in subsection 4.2.3.2 to yield the final Decision Matrix $\boldsymbol{D}^{(0)}$ for the iteration. Based on the first scheduling iteration ($p = 0$), the new mask $\boldsymbol{M'}$ is generated to exclude the now occupied inputs and outputs from the next scheduling iteration. In iteration $p = 1$, this mask is applied to $\boldsymbol{T}^{(1)}$ to generate the Assistant Matrix $\boldsymbol{A}^{(1)}$, which is processed by the RRPE and the *Sync* in the same way as in the first iteration. Finally, the current Traffic Matrix T is updated to

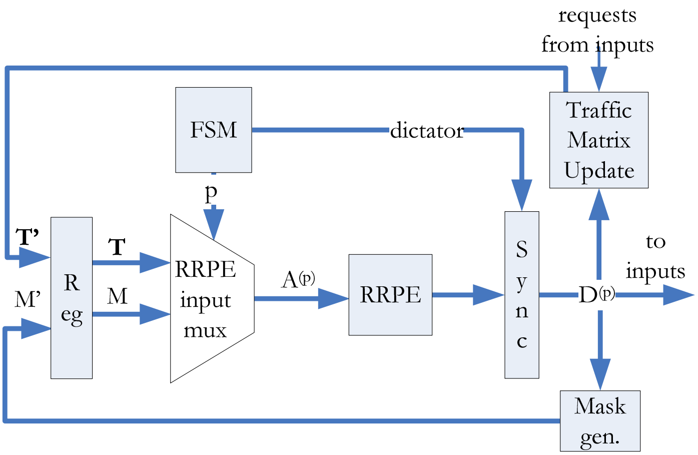

Figure 4.3 - MLRRMS hardware implementation. This scheduler performs look ahead using an iterative approach, where the fan-outs of the packets in the queues are scanned in a sequential manner, incrementing the position pointer p for each clock cycle.

represent the new state of the FIFO queues at the inputs after cell transmission $(\boldsymbol{T'})$ and the new mask $(\boldsymbol{M'})$ is reset so that all inputs and outputs will again be eligible for scheduling in the next run.

4.2.4.2. The ERRMS implementation

The ERRMS depicted in Figure 4.4 uses a parallel look-ahead mechanism to perform scheduling of both $p = 0$ and $p = 1$ in the same clock cycle, effectively doubling the scheduling speed of the multicast scheduler for a given clock rate. The main idea the proposed design approach is the realization, that an output will only have to search through $p = 1$ if the request vector for $p = 0$ is empty, i.e. if no HOL cells are destined for this output. Hence, by performing a simple OR operation on the HOL request row corresponding to the output $(\boldsymbol{T}^{(0)})$, it is possible to determine if scheduling should be performed on $p = 0$ or $p = 1$. Hence, only *one* RRPE is required to process both queue positions. The RRPE look-ahead MUX performs this operation and passes on an aggregated version of $\boldsymbol{T}$ to the RRPE consisting of a mix of $p = 0$ and $p = 1$ requests. The rest of the circuit works in much the same way as the MLRRMS, with three major differences:

- The mask generation logic is no longer required, since only *one* scheduling iteration is performed.
- The dictator and round-robin pointers are now shared between the two cell positions, as opposed to the MLRRMS, which used separate pointers for $p = 0$ and $p = 1$
- The *Sync* function works only on cells from $p = 0$.

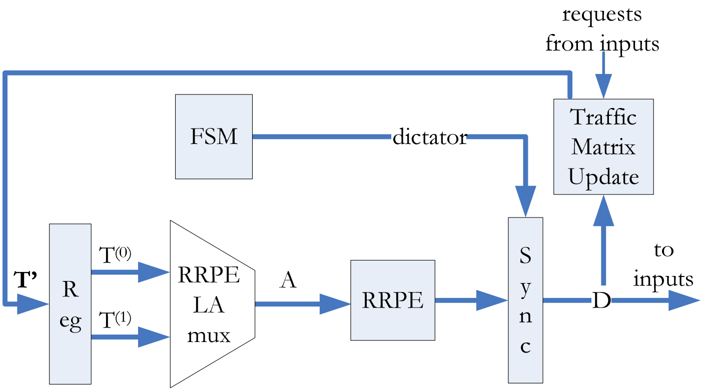

Figure 4.4 - ERRMS hardware implementation. This scheduler scans the fan-out vectors of the packets in position 0 (HOL) and position 1 in the queue and performs scheduling of packets in both positions in parallel. Hence only one clock cycle is needed.

4.2.4.3. The Round-Robin Priority Encoder

A key component of both of the schedulers is the round-robin priority encoder (RRPE). After considering several different implementations ([93], [94], [96]), the design described in [94] was chosen and modified slightly to form the circuit depicted in Figure 4.5, which is replicated for each of the N outputs. The main idea presented in [14] is to split up the priority encoding into two parts: The part from $x \in \{rr_{pointer}, \ldots, N - 1\}$ which is performed by PE$_{temp}$ and $x \in \{0, \ldots, rr_{pointer} - 1\}$, which is performed by PE$_{simpl}$. Both PE$_{temp}$ and PE$_{simpl}$ are simple priority encoders, each operating on their own subsection of the requests. The splitting is performed by first converting the rr-pointer to an N-bit mask such that the bits in position $0 \ldots rr_{pointer} - 1 = 0$ and $rr_{pointer} \ldots N - 1 = 1$. By performing a logical AND operation with the mask on the request vector, the result is a vector where all bits below the rr-pointer $(0 \ldots rr_{pointer} - 1)$ are equal to 0 and the rest keep the value from the request vector. By feeding this modified vector into PE$_{temp}$, this priority encoding will only be performed on the bits from $rr_{pointer} \ldots N - 1$. The input to PE$_{simpl}$ is just the unaltered request vector. The output from the two priority encoders are finally fed into a multiplexer, which selects the output from PE$_{temp}$ if this encoder has found a grant. If no grant is found in PE$_{temp}$ (i.e. if all the inputs to PE$_{temp}$ are zero), the output from PE$_{simpl}$ is selected. By selecting the two encoders in this prioritized order, the result is a priority encoder, which encodes the request vector with the priorities descending from $rr_{pointer}, \ldots, N - 1, 0, \ldots, rr_{pointer} - 1$.

4.2.5. Simulation Results

Simulations are carried out in OPNET Modeler® [98] to evaluate the performance of ERRMS compared to other multicast scheduling designs. To determine the effect of increasing the look-ahead depth for MLRRMS and ERRMS beyond the recommended value of $L = 1$, several values of

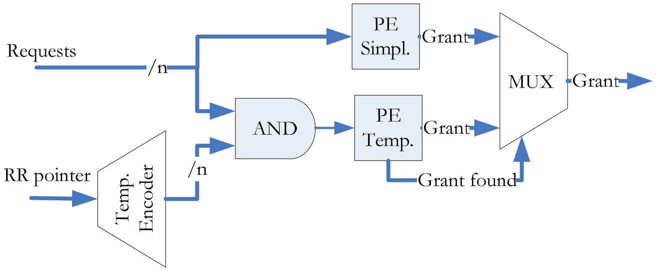

Figure 4.5 - The chosen Round-Robin Priority Encoder design performs RR encoding by splitting up the request vector in two halves (above and below the round robin pointer) while applying a mask to the lower half. The two parts are processed in parallel by standard static priority encoders and merged to produce the final grant.

L have been simulated. Independent multicast traffic is assumed to each input. To compare the performance of the algorithms in varying traffic conditions, Bernoulli traffic and bursty traffic with different fan-out schemes are considered.

4.2.5.1. Traffic Model

- Bernoulli traffic process: a cell arrives at an input with a probability of ρ. Thus the offered load can be calculated as $\lambda = \rho \cdot E[F]$, where F is the random variable of the cell fan-out.
- Bursty traffic process (*Correlated Arrival Process*): the process has two states, *Busy* and *Idle*. Cells are generated only in the *Busy* state. The process stays in each state for a random number of cell times following the geometric distribution with mean values of $E[B]$ and $E[I]$, respectively. The arrival rate is calculated as $\rho = E[B]/(E[B] + E[I])$, and the offered load can be calculated as $\lambda = \rho \cdot E[F]$. Since the traffic arrives at the switch in bursts, two modes of fan-out schemes can be applied, cell-based and burst-based.
- Cell-based fan-out mode: the fan-out vector is independently generated for each cell.
- Burst-based fan-out mode: the fan-out vector is independently generated for each *burst* of cells.

4.2.5.2. Performance Comparisons

Scheduling performances in terms of multicast delay are compared between WBA, FIFOMS, MLRRMS and ERRMS. Multicast delay is defined as the number of cell time periods that a multicast cell waits in the queue before being removed. A cell time period T is defined by the cell size s and the fabric byterate BR as $T = s/BR.$ Since fan-out splitting is applied, a multicast cell will not be removed from the queue until it is sent to all the destinations in the fan-out set.

Comparisons on the average multicast delay under Bernoulli traffic applied are shown in Figure 4.6. Under high load, the FIFOMS, as a benchmark, outperforms the others because FIFOMS uses the VOQ architecture to schedule multicast traffic by creating up to N tokens (N for the number of output ports) for each incoming packets and thus multicast HOL blocking is eliminated. The drawback of the FIFOMS is however the limited scalability as discussed. Since the WBA operates only on the HOL cells, it suffers from the HOL blocking and has the highest multicast delay as the output load increases. The ERRMS has better performance on the average multicast delay than the MLRRMS with $L = 1$ when the traffic load is heavy, and has the same performance as the MLRRMS with $L = 1$ when the load is light. This indicates that the ERRMS is able to provide efficient multicast scheduling performance with reduced HOL blocking. As ERRMS is expanded to higher values of L, its performance converges towards that of FIFOMS at the cost of extra hardware complexity. As such, ERRMS can be viewed as a more flexible and scalable alternative to the fully VOQ based FIFOMS system in terms of performance vs. cost. For higher values of L, MLRRMS performs slightly better than ERRMS in terms of multicast delay, but has the before mentioned limitations in terms of scheduling speed.

Having only Bernoulli traffic is not enough to evaluate switching scheduling algorithms since real-life traffic often comes in bursts. Thus the bursty traffic model is also applied. Comparisons on the average multicast delay under bursty traffic with cell-based and burst-based fan-out schemes applied are shown in Figure 4.7 and Figure 4.8, respectively. Due to the more uneven traffic distribution, it is observed that the switch becomes more unstable as the traffic load increases under bursty traffic than under Bernoulli traffic.

When a cell-based fan-out scheme is applied, as shown in Figure 4.7, the WBA is the first to become unstable because of the limited throughput caused by the HOL blocking. Since the FIFOMS eliminates the multicast HOL blocking, it results in the lowest average multicast delay. The ERRMS outperforms the MLRRMS with $L = 1$ as the output load increases, and significantly reduces the multicast delay compared to the WBA. As with the Bernoulli traffic, increasing the look-ahead depth L for

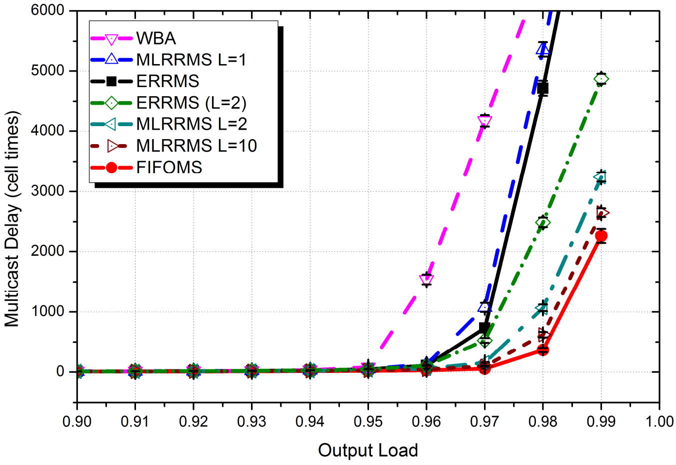

Figure 4.6 - Average multicast delay vs. output load, Bernoulli traffic, 95% confidence interval calculated on 30 seeds.

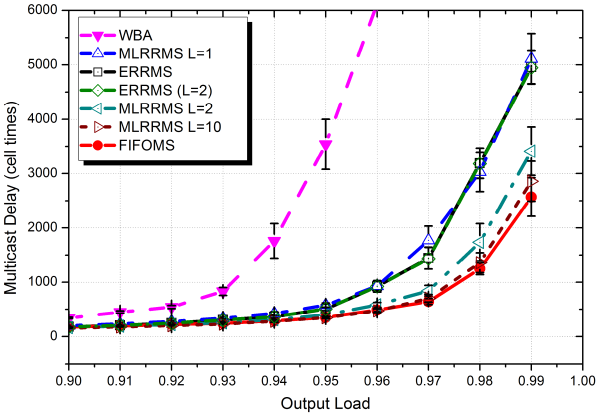

Figure 4.7 - Average multicast delay vs. output load, bursty traffic with cell-based fan-out scheme, 95% confidence interval calculated on 30 seeds.

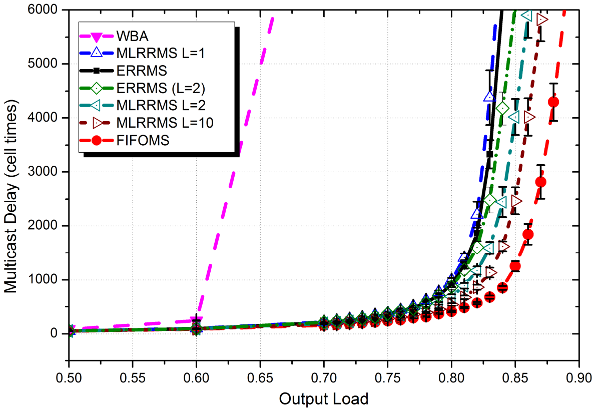

Figure 4.8 - Average multicast delay vs. output load, bursty traffic with burst-based fan-out scheme, 95% confidence interval calculated on 30 seeds.

ERRMS and MLRRMS offers only limited improvements to the multicast delay.

When the burst-based fan-out distribution is applied, as shown in Figure 4.8, cells carrying the same fan-out vectors arrive in groups and it results in a higher delay due to the uneven fan-out distribution in the queues. The ERRMS has slightly better performance than the MLRRMS with $L = 1$ as the output load approaches 0.85. Compared to the WBA, the ERRMS dramatically reduces the multicast delay. Increasing the look-ahead beyond $L = 1$ still only provides a small performance increase.

4.2.6. Hardware Synthesis Results

In order to compare the performance and resource utilization of the two designs, they have both been implemented and synthesized for an Altera Stratix IV FPGA (EP4S40G2F40I2) [42]. Both are configured to have a look-ahead depth of $L = 1$. As seen from the performance graph depicted in Figure 4.9, the ERRMS scheme outperforms the MLRRMS scheme with about a factor of two for all port configurations. Due to the efficient design of the ERRMS circuit, the maximum clock frequencies of the two designs are largely the same, even though the ERRMS processes two queue positions simultaneously. Consequently, the clock cycle count becomes the dominating factor in the processing speeds of the schedulers. Since the ERRMS needs only *one* cycle to perform a scheduling round, while the MLRRMS must process the two queue positions sequentially in *two separate cycles*, the ERRMS will perform approximately twice as many scheduling decisions per second as the MLRRMS.

The required speed of the scheduler is directly proportional to the supportable throughput of the fabric in cells per second, and hence, to the throughput in Gbps for a give cell size. For a 32-port fabric, the scheduling performance of the FPGA implementation will be able to perform 63 million scheduling decisions per second. This corresponds to sustaining a 32Gbps traffic steam using a cell size of approximately 64B, given perfect cell utilization.

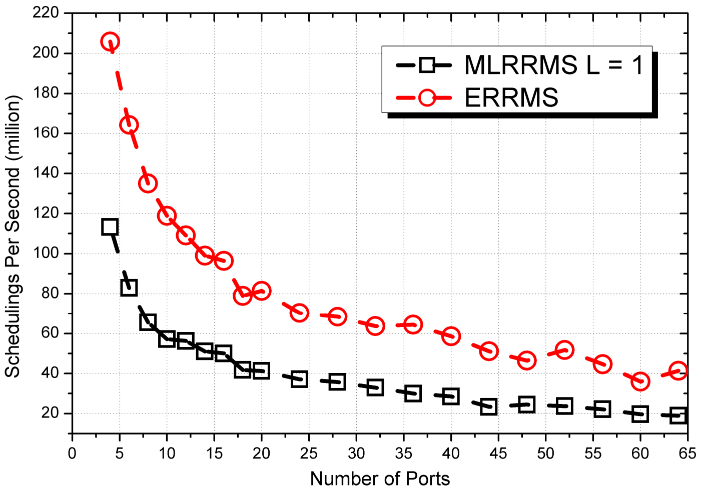

Figure 4.9 - Hardware synthesis results for an FPGA implementation: Scheduling speed vs. switch size.

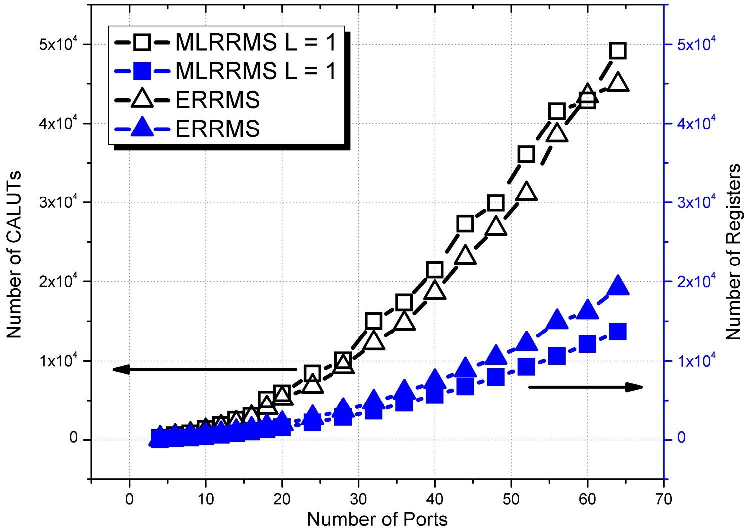

Figure 4.10 - Hardware synthesis results for an FPGA implementation: Scheduling speed vs. switch size.

Looking at the resource utilization of the two different schemes (Figure 4.10), it is clear that the two designs have very similar logic utilization (Combinational Adaptive Look-Up-Tables (CALUTs)) and register utilization for schedulers with less than 20 ports. As the number of ports grows beyond 20 ports, the ERRMS start using more registers than the MLRRMS, but consumes correspondingly fewer logic cells. Hence, the ERRMS does not cause any significant overhead on the gate count of the schedulers compared to the MLRRMS implementation even though it performs scheduling on two levels in parallel.

4.3. Chapter Summary and ERRMS Conclusion

This chapter has looked into the subject of switch fabric scheduling for input queued switches. An overall introduction has been presented, followed by an in-depth description of the ERRMS algorithm, a novel multicast scheduling algorithm with look-ahead ability, for IQ switch architectures. Instead of using the iterative method as in MLRRMS to perform the look-ahead searching for cells that can be sent to otherwise idle outputs, the ERRMS performs look-ahead in parallel. Thus, only one cycle is needed to perform one scheduling round. This enables ERRMS to be used in a high-speed switching environment. The synthesis result achieved for the FPGA based implementation indicates that an implementation based on modern ASIC technology or future FPGAs will likely be able to support the scheduling speeds required for e.g. 100 Gigabit Ethernet. The trade-off compared to MLRRMS is that the switch fabric and input buffers must be able to support transmitting two cells from one input simultaneously for the ERRMS algorithm to work properly. By taking the fan-out information of the first two multicast cells into account when scheduling, instead of just considering the HOL cell, the ERRMS is able to increase the sustainable throughput significantly compared to WBA while providing much better scalability compared to FIFOMS. Simulation results show that increasing the look-ahead depth beyond these two cells will only provide marginal improvements to the throughput performance. The synthesis results have shown that the ERRMS is capable of providing much faster multicast scheduling than the iterative MLRRMS for the implemented look-ahead depth of two cells, about a factor of two for all port configurations. The speedup is expected to grow in the same approximately linear fashion if the look-ahead depth is increased. Even though ERRMS performs scheduling on multiple levels in parallel, the logic size of the ERRMS scheduler is reasonable for hardware implementation and scales well to even very large switch fabrics.

5. High Performance Forward Error Correction

Forward error correcting (FEC) codes are an integral part of modern optical communication systems, as they allow for higher data throughput and longer reach (the so called bandwidth-distance product) for the same optical WDM channel width. As such, this was also a main research topic in the Celtic project “Elastic Optical Networks (EO-Net)” described in the introduction to this thesis[10]. This chapter presents the work done, as part of this project, to implement advanced FEC codes efficiently in parallel hardware to achieve high data throughput without sacrificing their error correction performance. The chapter first provides a short introduction to the two types of FEC codes used in the EO-Net project, with specific emphasis on the Low Density Parity Check Convolutional Codes (LDPC-CC). Furthermore, it introduces the concept of concatenating the two codes-types to enhance performance (Section 5.1). Secondly, it describes how implementations of these two types of FEC codes can be parallelized in hardware to improve throughput (Sections 5.2 and 5.3, respectively). The work presented in this chapter is targeted towards incorporation into designs running the Optical Transport Network (OTN) standard as described in Chapter 6, but it is generally applicable to other configurations as well.

5.1. Background

The hard-decision Reed-Solomon (RS) (255,239) FEC code is used for a broad range of applications, ranging from data storage to optical transmission systems. It is also the default FEC used for data correction in the OTN standard [99][100]. As depicted in Figure 5.1, this FEC makes it possible to transmit data reliably, at much lower signal-to-noise ratios (SNR) than without FEC coding. The cost is a 7% parity overhead on top of the information bits, corresponding to a code rate (information-to-transmission bitrate ratio) of approximately 0.9. However, since it is an algebraic code based on hard decision bits, the code gain provided by RS(255,239) is relatively low compared to more complicated codes[101]. To improve the noise resilience of optical transmission systems even further, the more advanced Low Density Parity Check (LDPC) soft decision FEC codes have been suggested[101]. LDPC is a so called

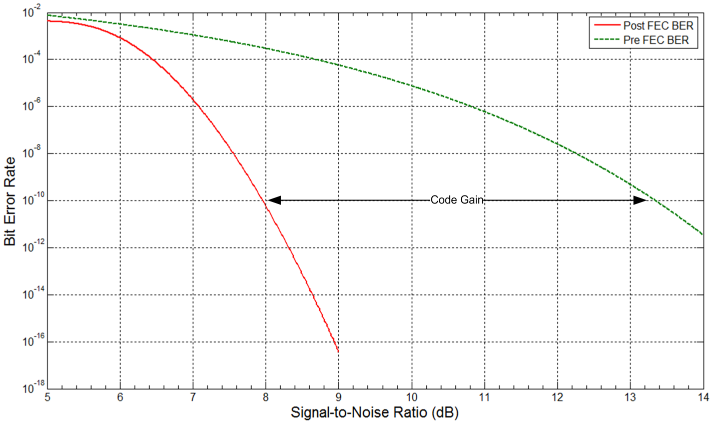

Figure 5.1 – Simulation showing the bit error rate before and after Reed-Solomon FEC decoding as a function of the SNR.

capacity approaching code, which means that it is able to achieve an error correction performance very close to the Shannon limit[102]. The price for the high performance is added complexity, especially in the decoder, compared to the standard RS FEC.

5.1.1. Low Density Parity Check Codes

An LDPC code of rate $R = k/n$ is usually defined by its parity check matrix $\boldsymbol{H}$, of size $[(n-k)\ \mathbf{x}\ n]$, where k is the length of the information sequence, and n is the length of the transmitted coded sequence. A bit sequence $\boldsymbol{c}$ is a valid (error free) codeword if $\boldsymbol{cH^T} = \mathbf{0}$, where $\mathbf{0}$ is an all-zero vector, and $\boldsymbol{H^T}$ is called the *syndrome former* of the code. LDPC codes are usually decoded using an iterative *message passing* algorithm, e.g. *sum-product* or *min-sum,* which work with *soft bits*. When soft bits are used, instead of simply using the received bit value ('1' or '0'), which may be in error due to channel noise, the decoder also includes information about the *reliability* of the bit. Soft decision decoding is shown to give a gain of up to 3 dB compared to hard decision decoding [100]. The reliability of the bit is usually represented by its *log-likelihood ratio* (LLR). The LLR L_b of a certain bit b is given by the log of the ratio between the probabilities of that bit being a '1' and a '0':

$$L_b = \log\frac{p(b=1)}{p(b=0)}.$$

The hard decision is obtained by taking the sign of the LLR : '1' if $L_b > 0$ and '0' if $L_b < 0$. By including soft information in the FEC calculations, soft decision LDPC is able to achieve code gains, much higher than standard hard decision codes such as RS(255,239) and can reconstruct information at much lower SNR values. LDPC codes are iterative codes, which means that they process the same bit sequences multiple times to improve performance, increasing the certainty of the LLR values for each iteration. After a number of iterations, a hard decision is made based on the sign of the LLR, and the binary bits are passed on to the next processing step in the receiver.

5.1.1.1. Convolutional LDPC Codes

In [103] the authors proposed a new class of LDPC codes, called *LDPC convolutional codes* (LDPC-CC). LDPC-CC rely on the periodic structure of the parity-check matrix $\boldsymbol{H}$ to accommodate practical encoding and message-passing decoding of arbitrary block lengths. In this subsection, the properties of the LDPC-CC are described regarding practical implementation. For a more comprehensive description and analysis of the error-correction capabilities of the codes, the reader is referred to [104], [103]. Convolutional codes, and thus LDPC-CC can be described by their memory M. The memory of the code shows how many previous bits the current encoded bit is dependent on. The work presented here is restricted to codes with rates $R = b/(b+1)$, where b is an integer, and memories $M = T + 1$, where T is the parity-check matrix period. First a $[(b+1)T \text{ x } T]$ low-density matrix (i.e. consisting of mostly zeros) is constructed randomly [100], after which the *unwrapping procedure* described in [103] is used to obtain the syndrome former $\boldsymbol{H^T}$. It is shown in [103] that the encoding can be performed by a shift register with variable connections to an output xor function as depicted in Figure 5.2. These connections are given in each clock cycle by the syndrome former matrix, which is stored in a local memory block and accessed in a cyclic manner with a period of T.

The choice of the code memory is crucial for its performance, as well as for the hardware implementation complexity and scalability of the system. In [103] it is proven, that the minimum distance of the code, and therefore its error-correction capability, grows linearly with M. However, increasing M requires increasing T, which leads to increased encoding and decoding

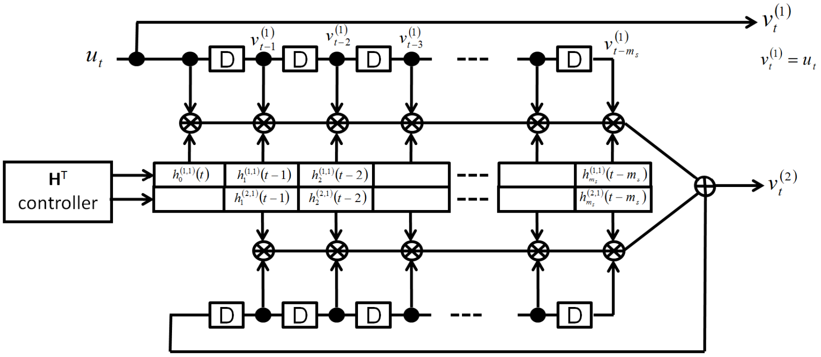

Figure 5.2 - Block diagram for a LDPC encoder of rate ½ [107].

complexity. A long period T means that the encoding circuit needs to be larger and the hardware memory requirements in the decoder is increased due to the necessity to store at least one period of the data in the system.

5.1.1.2. Terminating the LDPC-CC

Since OTN is a frame-based communication standard, each code block must be at most as long as the frame. This requires that the encoding process is stopped, once the desired length is reached, while ensuring that the encoded bit sequence $\boldsymbol{c}$ is still a part of the code space, i.e. $\boldsymbol{cH^T} = \mathbf{0}$. This process of ensuring code integrity is known as *termination.* Termination of a convolutional code usually means feeding the encoder a certain bit-stream, which sets the shift register to the all-zero state. However, since the connections here are flexible, and since the encoder has a feedback loop, this is not as straight-forward as for conventional convolutional codes. In [104], the authors derive a termination circuit for the LDPC-CC. However, it is rather complex for hardware implementation. In this thesis, it has been chosen to terminate the encoder by *zero-padding,* i.e. by adding a tail of zeros to the information seqeuence processed by the encoder. The length of the zero-sequence is known to the receiver, and thus only the resulting parity bits are sent on the channel along with the original codeword. This does not bring the shift register to an all-zero state, but if the zero-sequence is long enough, the performance degradation due to the improper termination is negligible. The zero-sequence used in this thesis is of length $M + 1$, which has been found to result in less than 0.1dB loss compared to the code with proper termination from [104]. Termination also results in a reduced code rate. If the information word length is k, and the non-terminated code rate $R = b/c$, the true code rate after zero-padding termination is $R_{true} = \frac{k}{\frac{c}{b}k+c(M+1)} < \frac{b}{c}$.

5.1.2. Concatenated FEC Codes

The LDPC codes have very good coding performance, approaching the Shannon limit. However, the LDPC codes generally show an error floor, which is too high compared to the common bit error rate target of optical transmission systems of at least 10^{-12}. One way of reducing this problem is to improve the so called girth of the LDPC code by increasing the codeword length or increasing the FEC overhead of the code. However, increasing the codeword length or the FEC overhead comes at the cost of logic complexity or reduced effective throughput, respectively[105]. A more efficient option is to use the well known concept of concatenated codes, as described in [105] for RS and LDPC codes. In a concatenated RS/LDPC code, the errors which are left uncorrected by the inner LDPC code are corrected by a second FEC layer employing the simpler Reed-Solomon code mentioned above. A RS (255,239) code is much less complex to decode than LDPC codes. Hence, the combined cost of using two FEC systems is significantly smaller than expanding a single LDPC FEC to achieve similar performance[105].

Figure 5.3 shows the output BER of an RS decoder as a function of the input BER coming from the LDPC decoder. As seen, the hard decision RS code works very well for low error rates and is able to further reduce the combined BER multiple orders of magnitude, going below 10^{-14} for an input BER of 10^{-4}. Hence, the goal output BER of the LDPC decoder will be around 10^{-4} to achieve practically error free transmission. This can be achieved by selecting an LDPC code appropriate to the SNR of the channel and by fine tuning the number of LDPC decoder iterations. These parameters should be selected such that the 10^{-4} BER target is met with the smallest possible resource requirements in terms of chip area, parity overhead and power consumption. This can be done either statically, based on worst case assumptions, or dynamically, as described later in Chapter 6, to continuously achieve an optimal trade off based on the current state of the system.

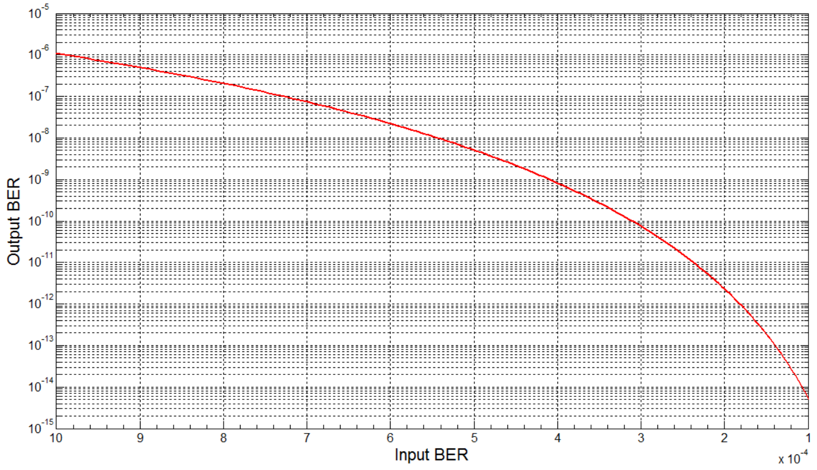

Figure 5.3 - Bit error rate after Reed-Solomon FEC decoding as a function of the input bit error rate assuming the worst case scenario of uncorrelated (non-bursty) errors at the input.

5.2. Reed Solomon FEC for OTN4

This section focuses on the Reed-Solomon (239/255) FEC algorithm specified by the earlier OTN standards [106]. It specifically examines and proposes solutions to the problem of scaling the algorithm to OTN4 line speed of 112Gbps, in an Altera Stratix IV FPGA. This FPGA family was state-of-the-art in 2010 when this research was performed [P4], and is also the FPGA type used in the demonstration platform for the EO-Net project[107]. The synthesis results included in this section show, that it is indeed possible to scale the algorithm to 112Gbps, using this FPGA technology with the proposed design approach.

5.2.1. RS FEC operation in OTN

The forward error correction (FEC) functionality is a key component, and one of the most logic resource consuming parts, of an OTN module. By means of the standard Reed-Solomon (239/255) FEC, the receiver is able to correct up to eight incorrectly received bytes in a 255B codeword, or to detect at least 16 incorrect bytes with no correction. The cost of this functionality is 16 extra bytes of data for each codeword. This means that for each 239B of user data, an additional 16B of FEC code needs to be transmitted, thereby totaling 255B. In order to improve the system's tolerance towards error bursts, the 16 codewords, which make up each row in the OTN frame, are byte-interleaved. This spreads error bursts over several codewords, thus decreasing the probability that any one codeword contains more than the eight byte errors, which are correctable by the algorithm. Since OTN frames consist of four rows, the total number of codewords in one frame is 64.

5.2.2. Parallel RS FEC implementation

The Reed-Solomon FEC algorithm in OTN is byte oriented [106]. Unfortunately, the standard 8-bit bus width is much too narrow to reach the 112Gb/s required for OTN4 using RS(239/255) FEC encoding. In order to reach this speed while working on 8-bits per clock cycle, the internal clock of the calculation circuit would have to run at 14GHz, which is highly unrealistic. To reduce the required maximum frequency to a reasonable level, a hardware based FEC solution has been designed and implemented, which processes 64B per clock cycle. The high throughput of the system is accomplished by processing each of the 64 individual FEC code words, which make up an OTN frame, in parallel. The 64 code words, each processed one byte at the time, add up to 512bits per clock cycle. This brings the required clock frequency down to 218.75MHz, which is a more realistic clock speed using current FPGA technology. The implementation is further facilitated by the practice of byte-interleaving code words, which makes it easier to distribute the data to the encoding circuits on the transmit side and the error correcting FEC decoder circuits on the receive side.

On the encoding side, the data for each of the four OTN rows is distributed to 16 individual encoding circuits (64 parallel encoders in total). After the encoders, the FEC data is multiplexed into the four byte-interleaved FEC blocks, which are transmitted at the end of each row (each containing the data from 16 encoders) as defined in the OTN standard [106]. At the receiver, a similar system could be implemented, consisting of 64 individual FEC decoding circuits, each working on a separate sub-stream of bytes. These systems would all consist of a syndrome calculator, a polynomial division circuit and an error amplitude/position calculation circuit as depicted in Figure 5.4. Unfortunately, the combined circuitry of the correctors is far more complex than the simple encoding circuits, making this approach an unviable option from a resource utilization perspective.

Luckily, the polynomial division circuit (PDC), which is the most complex and resource consuming part of these decoders, has a significantly higher throughput than the other components. The PDC only uses a maximum of 16 clock cycles to process an entire 255B codeword, whereas the other two components use 255 cycles for their operations. As a consequence, one PDC can handle the data from up to 16 parallel FEC calculations, if they are clocked at the same speed as the rest of the system. Hence, the number of PDC circuits in the system can be reduced to four, if a simple resource sharing system is implemented. The parallel decoding circuit is depicted in Figure 5.5. In order to reduce the clock constraints on the PDC to meet the limitations of the FPGA for which the design was targeted, the synthesized design uses five PDC circuits. This reduces the PDC clock to 164.5MHz at the cost of an additional PDC and some slightly more complex multiplexing and demultiplexing circuitry on each side of the PDCs. In modern FPGAs this would not be necessary due to their faster logic speed.

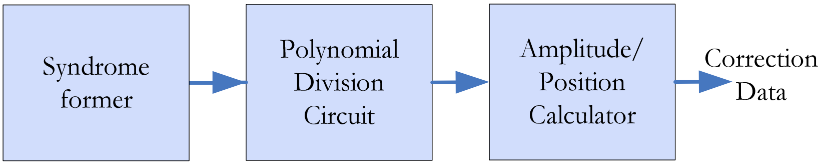

Figure 5.4 - Serial Reed-Solomon decoding circuit.

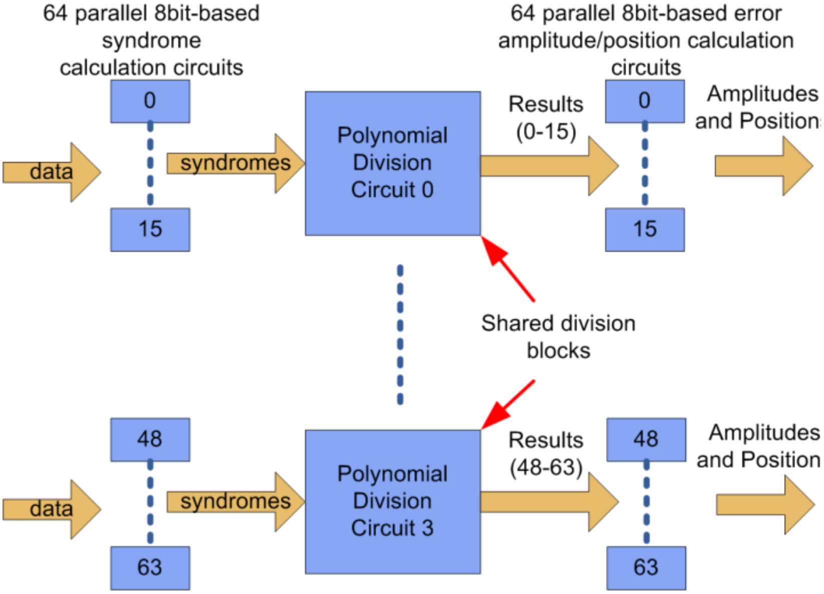
64 parallel 8bit-based syndrome calculation circuits
64 parallel 8bit-based error amplitude/position calculation circuits
0
15
data
syndromes
Polynomial Division Circuit 0
Results (0-15)
0
15
Amplitudes and Position
Shared division blocks
48
63
data
syndromes
Polynomial Division Circuit 3
Results (48-63)
48
63
Amplitudes and Position

Figure 5.5 – Parallel Reed Solomon decoding circuit.

5.2.3. Synthesis Results

In order to verify that the FEC algorithm can be implemented to run fast enough in a single FPGA, the system has been synthesized to a medium sized Stratix IV GT FPGAs from Altera (EP4S100G2-F40I2), which is designed for 100Gbps applications [42]. The synthesis results for the individual components of the RS implementation are listed in Table 5.1. Using this FPGA, the proposed design takes up approximately 46% of the FPGA resources, which still leaves plenty of room for the surrounding control logic, which is necessary to complete the system. The resource consumption of the proposed system is relatively large, but significantly smaller than a fully parallelized design (using 64 separate decoders) which uses over 171% of the FPGA resource. The resource usage of the system is still relatively large, but not unexpected given the high throughput requirements. For comparison, a 10Gb/s circuit would take up around 5.41% of the FPGA resources, while running at 208.33MHz using the same resource sharing approach. Using the FPGA from the EO-Net demo board, which has the highest number of logic elements available in the Stratix IV family, the proposed 112Gbps design would only take up approximately 20% of the logic resource.

	64 Enc.	64 Syndr. Calc.	5 Poly. Div.	64 A/P Calc.	Total
Logic Util.	6%	4%	11%	25%	46%
Comb. ALUTs	10,761	8,193	19,440	44,736	83,130
Registers	8,704	8,192	5,195	18,048	40,139
Fmax (MHz)	388.95	353.61	203	237.19	

Table 5.1 - Synthesis results for medium sized Stratix IV FPGA (182,400 Comb. ALUTs).

5.2.4. Reed Solomon Summary

This section has outlined the research challenges of scaling the standard OTN RS(239/255) Forward Error Correction algorithm [106] to 100Gb/s speed. Based on this, it has presented a viable solution to obtaining the required throughput using a Stratix IV FPGA. In particular, it is shown that the necessary processing speed can be obtained, at a realistic clock frequency, by means of parallelization. The results show that it is indeed possible to implement this algorithm for 100Gb/s operation, as part of a larger OTN-4 system, using only a fraction of the available FPGA resources. With modern FPGAs being both faster and larger (more than twice the number of logic elements is announced for Aria 10), the proposed design will even scale to 448Gbps if a future OTN-5 standard still uses Reed-Solomon (255,239).

5.3. Parallel LDPC-CC for high speed connections

As mentioned in Section 5.1, soft decision LDPC codes are much more complicated in terms of calculations per bit compared to hard decision algebraic codes like the Reed-Solomon (255,239) described above. As part of the EO-Net project it was thus investigated how best to implement an LDPC-FEC circuit, which was both fast enough to reach the 14Gbps speed of the FPGA-based proof of concept platform, while allowing for variable code rates[10]. The initial LDPC-CC encoder and decoder circuits built for the project were only capable of reaching 1200Mbps and 20Mbps respectively in a Xilinx Virtex 6 FPGA, and a trial synthesis for the Stratix IV FPGA used in the EO-Net prototype setup has shown similar results

[107]. This section will show how these designs can reach gigabit speeds by parallelization.

5.3.1. Encoder

The serial encoder is by far the faster and the smaller of the two circuits. For this reason, the proposal here is to simply replicate the encoding circuit multiple times to form 16 encoded streams in parallel. These streams will then be multiplexed to form the high speed serial output stream. As will be described in the synthesis results section (5.3.3), the aggregated throughput of the encoders can reach over 21Gbps, while using only 2-3% of the logic resources available on the Stratix IV FPGA.

5.3.1.1. Parity matrix compression

It may be noticed from the synthesis results, that the encoders do not use any block memory resources, even though each encoder has an H^T matrix of size $[cT \text{ x } T]$ attached to it (as described in subsection 5.1.1). This is because the matrixes are implemented as distributed memory, i.e. using logic elements. Because of the sparse nature of the parity matrixes (hence the name Low Density Parity Check Codes), it was found to be highly inefficient to use regular memory blocks to store the matrix, since most of the memory would be occupied by zeros. By specifying the memories as distributed memory, the synthesis tool is able to use Boolean optimization techniques to effectively compress the 64kbit of data (for one encoder) into only 403 ALUTs. For higher values of T, the compression factor is even higher. To further optimize the circuit, one could consider using just a single H^T matrix to drive all 16 encoders in unison. The extra path delay between the shared matrix and the encoders is easily reduced by pipelining the memory access.

5.3.2. Decoder

When it comes to the decoder, the lower speed and much higher memory requirements mean that it is impossible to reach multi-gigabit speeds in the Stratix IV FPGA by merely duplicating the circuit. The resulting design would simply be too large to fit in the FPGA. Here, it is absolutely necessary to perform internal parallelization of the decoding circuit, before it is replicated. Before moving onto the issue of parallelization in subsection 5.3.2.2, subsection 5.3.2.1 provides a condensed description of the decoding algorithm, and establishes the terminology used throughout the rest of this chapter.

5.3.2.1. Basic Implementation Aspects of LDPC-CC Decoding Circuits

The LDPC decoding is performed by a number of processors, each performing *one* iteration of a message-passing algorithm (the *min-sum* algorithm in this case). As opposed to the LDPC Block Codes (LDPC-BC), LDPC-CC decoders do not require each iteration to complete before the next one can start. The data in LDPC-CC decoders can flow continuously through a series of iteration processors, each working on a separate data window. This is possible because data dependencies within the algorithm can only reach M bit positions back in time. The decoding within the processors is best illustrated using a Tanner Graph based on Check Nodes (CN) and Variable Nodes (VN) as depicted in Figure 5.6. The Tanner graph shows the xor relationship over time between the coded bit sequence, as defined by the encoder and the H matrix for the code. Hence, compared to the encoding circuit depicted in subsection 5.1.1, Figure 5.2, the VNs correspond to the shift registers (D) and the CNs correspond to the right-most xor which generates the output parity bit. At each point in time, the algorithm checks the VNs against the original xor equations and updates the LLR values of the VNs correspondingly. Once a bit from the channel has been processed through the entire window, the LLR contents of the VNs corresponding to that particular bit are used to calculate an updated LLR value, which is passed on to the next processor. In hardware implementations, the algorithm can be split into a Check Node Unit (CNU), which compares and updates the LLRs of the current VNs, a Variable Node Unit (VNU), which updates

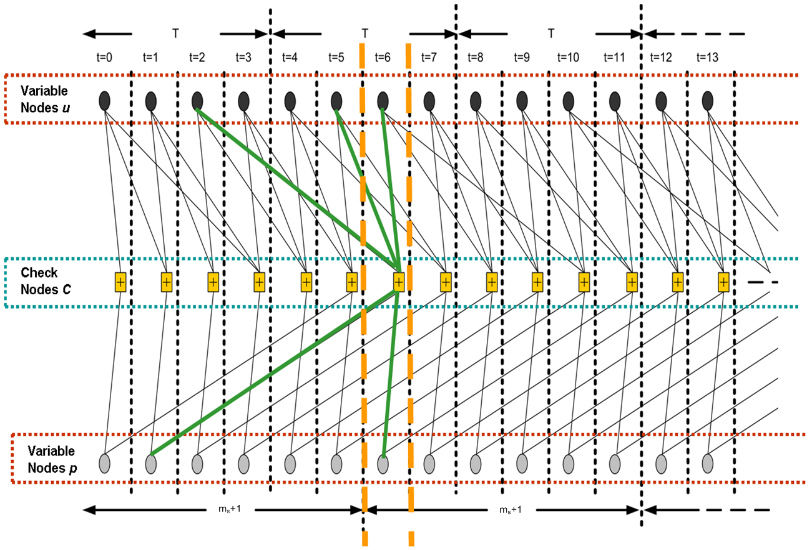

Figure 5.6 - Tanner Graph for rate ½ (5, 3(2), 5) LDPC-CC time-variant decoder. Each processor deals with a specific ms+1 part of the graph. Black and grey nodes represents variable nodes with three (3) and two (2) variable degrees respectively and yellow nodes represent check node units with five (5) check degree. [107].

the LLR of the next bit to exit the iteration, and some memory to store the values of the VNs as they are being processed. Furthermore, some control and switching circuitry is required to connect the CNU to the correct VN memory locations at each point in time. As described later, parallel implementations may use multiple instantiations of these components to increase throughput.

5.3.2.2. Decoding in parallel

The original decoder is already parallelized in the iteration domain by separating each iteration of the code into separate sub-processors, which can run in a pipelined fashion. However, the individual check node and variable node updates, corresponding to the processing of a single data word, are performed sequentially, as are the memory accesses. For the check degree of 12 used in this implementation, each CN is connected to 12 VNs. Hence, 12 memory read and 12 memory write operations must be performed for each CNU calculation. The VNUs use one cycle to write the arriving word to memory and four cycles to read the four VNs, which produces the output word at the edge of the processor window. Consequently, each data word takes around *29* cycles to process [107]. By using dual-port memories for the Variable Nodes (VN) and parallelizing the Check Nodes Update (CNU) and VN Update to perform all 12 calculations and updates in parallel, this can be reduced to just 2 cycles. To allow for this parallel operation, the circuit must distribute the VNs over 12 different memory banks and construct the code in such a way that the individual data banks are accessed exactly once per update. The modified circuit is depicted in Figure 5.7. These modifications bring the

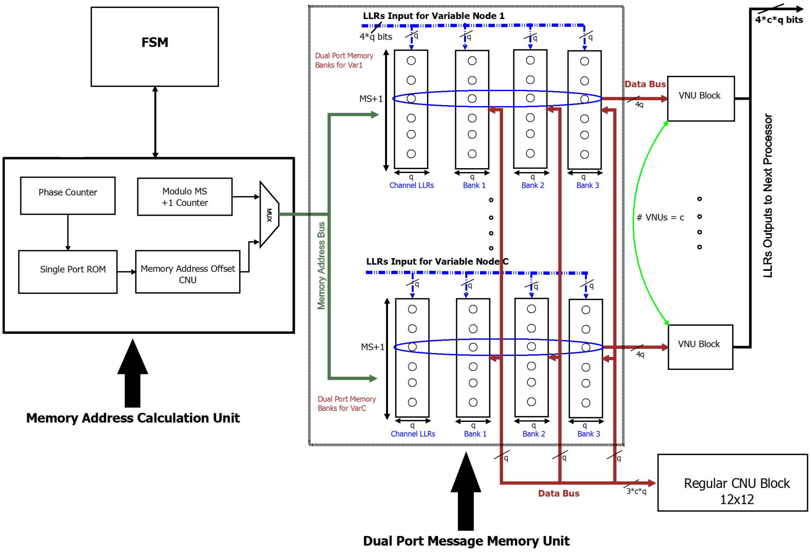
FSM
Phase Counter
Modulo MS +1 Counter
MUX
Single Port ROM
Memory Address Offset CNU
Memory Address Calculation Unit
Memory Address Bus
LLRs Input for Variable Node 1
4*q bits
q
Dual Port Memory Banks for Var1
MS+1
Channel LLRs
Bank 1
Bank 2
Bank 3
Data Bus
4q
VNU Block
# VNUs = c
LLRs Input for Variable Node C
Dual Port Memory Banks for VarC
MS+1
Channel LLRs
Bank 1
Bank 2
Bank 3
4q
VNU Block
q
Data Bus
3*c*q
Regular CNU Block 12x12
Dual Port Message Memory Unit
4*c*q bits
LLRs Outputs to Next Processor

Figure 5.7 – Semi parallel processor architecture [107].

throughput of the decoding circuit up to around 190Mbps in the Stratix IV FPGA.

To further increase the throughput, the decoder has been modified to process multiple input words, i.e. steps in the Tanner graph, in parallel based on the approach described in [108]. Aside from the necessary hardware modifications, processing words in parallel also puts constraints on the construction of the LDPC-CC code itself. In the serial case adjacent words can be interdependent, i.e. the correction of one word may effect the correction of the neighbours. In the parallel case, the word interdependence must be restricted, such that words which are processed in parallel do not affect one another. When this constraint is fulfilled, the data can be processed in parallel. The parallel system for a parallelization factor of $P = 4$ is depicted in Figure 5.8. As can be seen, the parallel system has P CNU units, which each processes the 12 VN values corresponding to a single input word. To ease the hardware implementation, the code has been constructed in such a way that the 48 VN values for the 4 adjacent steps in the Tanner graph are stored in the same 12 physical memory locations, each of which contain 4 values. The interconnection between the VNs and the CNUs are defined in a separate connection memory, which controls a switching network. The switching

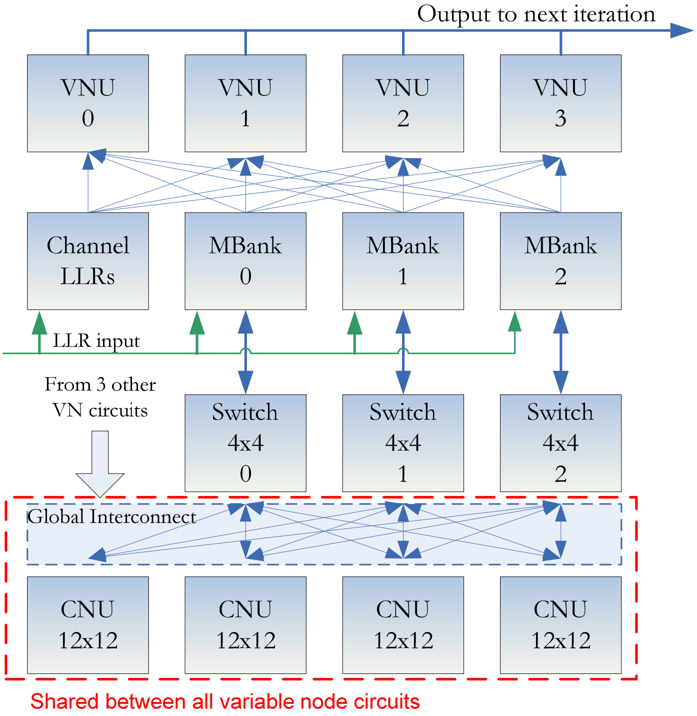

Figure 5.8 – Fully parallel decoder (only logic for one out of four variable nodes shown). Memory address calculation unit has been omitted for simplicity.

network divides the 48 values amongst the 4 CNUs, according to the current position in the Tanner graph, and makes sure that the outputs of the CNUs return to the corresponding VNU memory locations.

5.3.2.3. Forming the parity matrix

The parallel structure of the decoder puts several restrictions on the construction of the parity-check matrix:

1. The 4 CNUs work on 4 data words in parallel in each update cycle. Hence, there can be no data dependencies between VNs updated by different CNUs within the same 4-word window.
2. To reduce logic complexity, the distribution of LLRs over the 4 CNUs is performed using 4x4 barrel shifters, so these connections can only be linear shifts. This structure also ensures that restriction (1) holds. The parity-check matrix must reflect this restricted VN-to-CN relationship.
3. Furthermore, for the decoder to function the VNs must be distributed over 3 banks in the decoder. This distribution must be made such that the VNs corresponding to each 4-word window are stored in three different banks, with exactly 16 LLR values in each, to form 12 inputs to each of the CN. That way, no bank needs to be accessed more than once for each CN operation. The same restriction is applied for the VNUs, which also requires 48 inputs from the three memory banks.

The parity-check matrix is generated at random using Matlab with the above restrictions in mind. The rows and the columns correspond to the CNU and the VNU connections, respectively, and are assigned 4-by-4 because the circuit processes 4 words in parallel. As the matrix is generated, the VNs are distributed over the three banks to allow for the parallel access described above. The parts of the parity-check matrix corresponding to each VN circuit are generated separately. Below follows the pseudo-code for generating the matrix:

```
For each VN circuit
    For each column in $H_{VN}^{T}$
      While used banks <3
            Select random unused row R
            Select random bank number B which is unused in this
            row and column (see 3 above)
            Select random rotation Z (for restriction 2 above)
            Insert the 4x4 Z rotated matrix with bank B in row R
      End While
    End For
End For
```

VNU CNU	t=0	t=1	t=2	t=3
t=0	1000 0100 0010 0001	0000 0000 0000 0000	3000 0300 0030 0003	0002 2000 0200 0020
t=1	0020 0002 2000 0200	3000 0300 0030 0003	0100 0010 0001 1000	0000 0000 0000 0000
t=2	0000 0000 0000 0000	0100 0010 0001 1000	2000 0200 0020 0002	0003 3000 0300 0030
t=3	0030 0003 3000 0300	0020 0002 2000 0200	0000 0000 0000 0000	1000 0100 0010 0001

Table 5.2 – Example of a 16x16 decoder party matrix. Numbers 1-3 corresponds to a connection in banks 1-3.
Zero = no connection.

There is a chance that the algorithm may deadlock towards the end of the matrix, in which case the matrix construction (for this VN circuit) is redone from scratch. For a 16x16 matrix, corresponding to a period $T = 16$, the result for one VN can look like Table 5.2. Based on this matrix, the encoding matrix is easily generated by inserting a '1' instead of the bank numbers. For the decoder, the information for each point in time $t\epsilon[0..T/4 - 1]$ in the period of the code is stored in separate memories controlling the 4x4 switches (rotation) and the memory access (address/bank pairs).

5.3.3. Synthesis Results

As seen in Table 5.3, the modified decoding circuit with the described parallelizations, has a significantly higher throughput than the serial version. The parallellized decoder reaches 664Mbps, while using only 5% of the available look-up tables (LUTs) and less than 1% of the available register and block memory (BM) resources. With this reduction it is now feasible to replicate the circuit to handle the 16 streams from the 16 encoders. The aggregated throughput in the target FPGA is now 10.624Gbps using 82% of the available logic resources and less than 5% of the internal SRAM memory. It should be noted, that both the speed and the logic density of FPGAs are significantly lower than competing platforms such as ASICs. It is therefore expected that the design will scale to 100Gbps in an ASIC implementation. Furthermore, as previously mentioned, current FPGAs are both larger and faster than their older siblings, which are used in this project.

	Throughput(1)	Resources % (1)			Throughput(16)	Resources % (16)		
		LUT	Reg	BM		LUT	Reg	BM
Encoders	1.346 Gbps	0.14	0.2	0	21.536 Gbps	2.24	3.2	0
Decoders	0.664 Gbps	5	0.6	0.3	10.624 Gbps	80	9	4.6

Table 5.3 - Synthesis results for implementation in Stratix IV FPGA (EP4SGX530NF45C3).

5.4. Chapter Summary

This chapter has given an introduction to the two types of FEC codes used in the EO-Net project and shown how they can be implemented in parallel hardware for high throughput systems. The Reed-Solomon code is easily parallelized to reach over 100Gbps in a Stratix IV device using less than 20% of the available resources in the largest FPGA in the family. For the LDPC-CC code, parallelization is more complicated. Not only is the algorithm itself much more complex, but the parallelization opens up for memory access collisions and restrictions on data dependencies. This must be thought into the parity-check matrix when defining the code, as well as the hardware design. The proposed LDPC-CC design runs at 10.624Gbps using 82% of the logic resources available in the Stratix IV FPGA. Combined with a RS 10Gbps encoder, the total resource usage becomes 87% for the concatenated FEC (not including control logic and interleaving circuitry between the two FECs). Consequently, it may be possible to implement a complete 10Gbps concatenated FEC on the prototype platform from the EO-Net project. This can be used for optimization and proof of concept, before the design is ported to a more high performance FPGA or ASIC platform.

6. Energy Efficient Optical Transport Networks

While the previous chapter focused on enhancing Forward Error Correction (FEC) throughput and performance for next generation optical links, this chapter addresses the issue of improving energy efficiency. The chapter provides a general overview of how elastic optical networks, which are the focus of the EO-Net project, can reduce overall power consumption and moves on to examine adaptive FEC in greater detail, as a means of energy conservation.

6.1. Overview of Elastic Optical Networks

The concept of dynamic data rate adaptation is widely used in modern wireless transmission technologies, where spectral efficiency is traded off for higher reliability on low quality links by changing parameters such as the modulation format or the code rate of the FEC [109]. On optical links, on the other hand, the trend has commonly been to use fixed data rates due to the more static conditions of the underlying transmission medium. However, in recent years the concept of *elastic optical networks* has received increasing attention as a means of reducing the overall power consumption of optical networks[8], [110]–[112]. The main idea is to dynamically adjust the links depending on the current capacity demand to the most power efficient configuration which will still provide the required bandwidth. Of the several parameters which can be modified to reduce power consumption at the cost of throughput, the EO-Net project has primarily focused on the following:

- *Reducing the number of active channels:* This is one of the more efficient ways of reducing power consumption as all the dedicated optical and electrical equipment for the deactivated channels can be put in low power mode or turned off completely. However, opening and closing a channel takes time and can cause disruption in the traffic running on the link.

- *Reducing the per channel optical symbol rate:* This method means less load on the electronic components, which needs to process the data such as the digital signal processors (DSP) and the FEC. The downside is a possible loss of synchronization in the receiver as the change is taking place.

- *Using less spectrum efficient modulation schemes:* As known from general information theory, less spectrum efficient modulation schemes can function at lower signal to noise ratios (SNR) for a given bit error rate (BER)[113]. Hence, less regeneration, amplification and signal processing will be needed. As with the symbol rate, the digital electronics such as the FEC can also run at a lower rate thus saving power. Like the two previous techniques, changing the modulation will usually cause a temporary disruption to the link until the DSP algorithms converge to the new modulation format [114], [115]. Yet, recent work has shown that it is indeed possible to change modulation formats between optical packets without disruption at the cost of a small amount of extra overhead before each packet [114], [115]. It would also be possible to retain the same channel throughput with a less efficient modulation format if the spectrum was allowed to overlap adjacent (free) channels in the fiber. Thus, shutting down *one* active channel could not only save the energy for that channel, but for neighbouring channels as well, if they can utilize the released spectrum capacity. However, since the current ITU standard requires each channel to stay within a fixed 50GHz frequency band, this feature would require a more flexible channel grid as described in [8].

- *Adding more FEC overhead:* Inserting extra parity bits at the cost of lower effective throughput will increase the FEC's ability to correct errors and thus provide a coding gain corresponding to increasing the SNR. As a consequence, less regeneration, amplification, signal processing and FEC decoding iterations are needed to achieve the same BER. The FEC overhead can be changed without disrupting the traffic if the decoder knows *exactly* where in the bit stream this change is taking place – given that the FEC is designed with this functionality in mind.

While all of these parameters are good candidates for saving power, there is a clear difference in how often the different parameters can be changed in practice as well as the potential power reduction from changing the parameters.

6.2. Adaptive forward error correction

This section focuses on using the FEC overhead as a mean of dynamically trading off bandwidth for power reduction. The aim is to use the extra overhead to reduce the power consumption of the FEC circuitry itself. The scheme leverages the fact that modern iterative decoders will need fewer decoding iterations, and thus less power, to restore the data as the parity-to-information bit ratio increases. As mentioned in Section 6.1, an advantage of manipulating the FEC is that this can be done with no ill effect to the live traffic on the link, since the physical part of the transmissions system is independent of the FEC format. Also, with the right implementation, the rate resolution can be made very high with limited impact on the hardware complexity of the FEC. For these reasons, the code rate is an excellent parameter for utilizing small and brief traffic variations, i.e. the ripples in the capacity demand, to conserve energy. Even though the immediate power reduction will be relatively small compared to e.g. turning off an entire WDM channel, the effect will accumulate as it can run continuously on top of other power reduction measures. Hence, it would provide a small, but continuous reduction in power consumption. In order to facilitate seamless integration into existing systems, the proposed scheme is designed as a transparent add-on to transceivers running the Optical Transmission Network (OTN) protocol [99] and is fully backwards compatible. Hence, this proposed extension can easily coexist with current OTN implementations in a network. The rest of the section is structured as follows: Subsection 6.2.1 gives an overview of related work in the field of adaptive FEC for optical networks. Subsection 6.2.2 gives a brief introduction to the challenges related to building a rate adaptive FEC circuit. Subsection 6.2.3 describes how the OTN standard is modified to allow frame-by-frame rate adaptation and Subsection 6.2.4 describes how manipulating the rate can reduce the power consumption of the FEC decoder. Subsection 6.2.5 presents simulation results, which indicate how this scheme will affect the power consumption of an optical transmission system for a specific subset of FEC codes with different code rates. Section 6.3 concludes on the work presented in this chapter.

6.2.1. Related Work

In recent years, there has been increasing interest in the idea of employing the familiar concepts from adaptive radio communication to optical links. In [111] an adaptive FEC for optical communication systems was proposed which was later expanded in [112] to include adaptive modulation. The FEC adaptation is done by using separate FEC algorithms for each rate, which give the optimal FEC performance, although it is more costly in terms of hardware complexity. The main goal of the scheme is to automatically trade off transmission bandwidth for optical reach by using the measured BER at the receiver to select the appropriate FEC/modulation combination. The authors assume that there exists a feedback path to relay this information back to the transmitter. This scheme works well for finding the most efficient static parameters for an optical link, but lacks the capacity to perform fast adaptation without packet loss since it is not designed for dynamic power reduction. The work in [101] describes the benefit of using Soft- Decision (SD) FEC codes, specifically LDPC codes [116], to improve the reach of long haul optical networks, but also highlights the heavy complexity and high power consumption of the iterative soft decision decoding. To reduce the overall power consumption of the network, the authors suggest to adapt the number of decoding iterations based on the BER performance which comes from the quality and length of the individual links. Using fewer iterations results in lower coding gain but reduces the processing load on the FEC decoder. Using Dynamic Frequency and Voltage Scaling (DFVS) this translates to a corresponding reduction in the power consumption of the decoder. In a simulated US network, the authors show a power reduction of up to 82% in the FEC decoders compared to using the worst case number of iterations for all links. As with the rate adaptive FEC described in [111] and [112], this scheme is also focused on optimizing the FEC of a static link. However, the main idea of reducing the number of iterations based on the post-FEC BER is well suited for elastic transmission systems where the BER performance can be increased at the cost of decreased throughput. This is explained further in subsection 6.2.4.

6.2.2. Rate Adaptive Channel Coding

The system proposed for the adaptive FEC is a concatenated design, using a combination of the simple hard decision Reed-Solomon code (RS), combined with a soft decision LDPC-CC code as described in Chapter 5. The overall idea is to keep the RS system static, while manipulating the LDPC-CC code (or leaving it out altogether) to increase the overall code rate. The most straightforward approach to changing the code rate of the LDPC system is to use different codes for each required code rate as proposed in [112]. The benefit of this approach is that each code can be tailored to provide the optimal BER performance at the given rate. The drawback is that the hardware will need to support multiple different codes, potentially requiring a separate encoding/decoding circuit for each code, which increases the hardware complexity of the FEC. A different approach to support a large number of code rates with minimum hardware complexity, is to implement just *one* mother code of medium rate and then use *puncturing* and *shortening* to respectively increase and decrease the code rate at the output. Puncturing works by selectively removing parity bits from the encoded frame before transmission and inserting erasures, i.e.

bits with zero LLR values, before decoding at the receiver. Shortening, on the other hand, reduces the number of information bits included in the frame, replacing them with known "dummy bits" at the encoder and decoder with maximum LLR values. Since the min-sum is very sensitive to puncturing/shortening, this method would require very careful design of the puncturing and shortening patterns.

To get the best possible variable code performance, while keeping the hardware overhead to a minimum, the design proposed and simulated in this thesis uses a system with an SRAM defined encoder/decoder circuit. Such a system would use the same basic logic blocks to dynamically change the syndrome former to the one for the desired code rate. This is done simply by manipulating the connections from the shift register, based on the content of the SRAM. In this way, the codes for each rate can be highly optimized since they do not have to rely on the same base matrix. The desired sub-code is selected by manipulating the upper address bits of the SRAM memory. The exact design and implementation of the dynamic encoder/decoder is out of the scope of this thesis.

6.2.3. Frame-by-Frame Rate Adaptation

In order to perform truly gapless code rate adaptation the receiver needs to know exactly where in the bit stream the new code rate starts. For this purpose out-of-band signalling such as it is described in [111] is insufficient. Instead the proposed system transmits this information along with each optical frame. Not only will this approach enable gapless code rate transitions over a single link. It also makes it possible to use this scheme in packet switched optical networks, where the incoming packets may originate from different sources and have travelled along different light paths, thus requiring different code rates to reliably reach the receiver. The following subsections describe how this extra information is encoded into the standard OTN frame format and how the system as a whole can be modified for rate adaptive soft decision codes, while retaining backwards compatibility with existing OTN equipment.

6.2.3.1. Frame format

Figure 6.1 shows the basic frame format used by the OTN standard. The first 6 bytes of the frame contain the Frame Alignment Sequence (FAS), a fixed pattern which is used to detect frame boundaries in the serial bit stream. To convey the code rate of the frame to the receiver, the rate adaptive system adds an additional Code Identifier (CI) field immediately after the FAS. The new frame format is depicted in Figure 6.2. It has the same overall format as the standard OTN frame, aside from the CI field and a variable amount of soft decision FEC bits, which can be placed at the end of the frame or distributed over the entire frame depending on the type of SD code used. Hence the size of the outer frame (inner frame

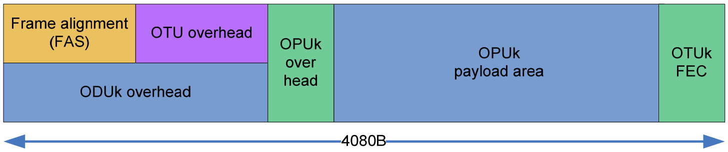

Figure 6.1 – Standard OTN Frame structure (not to scale) [99].

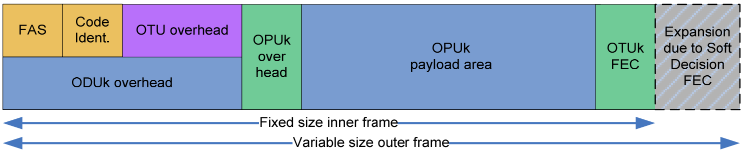

Figure 6.2 – Proposed variable rate frame structure. Here the FAS field is followed by a special code iden-tifier (CI) field. The length of the frame depends on the inner FEC code, which can be determined from the CI.

+ SD FEC bits) depends on the selected code rate. It has been decided to keep the outer frame size variable in order to ensure that each outer frame contains exactly *one* fixed size inner OTN frame, thus easing the demapping procedure at the receiver and eliminating the need for an internal frame alignment field. The downside to this approach is that the distance to the next FAS and CI fields must be derived from the current CI. As a consequence, an error in the CI field will not only affect the current frame, but may also cause the receiver to lose frame alignment. To avoid this problem it is imperative that the information in the CI field is interpreted correctly with overwhelmingly high probability, even at very high bit error rates. This is of special concern since the CI field is not covered by the FEC. This issue will be addressed in the following subsection.

6.2.3.2. Rate header encoding

In order to make the system resilient to bit errors, it is necessary to protect the CI from bit errors. This is done by means of a repeated coset of an extended (8,4) hamming code[100]. The hamming code is based on the standard OTN FAS, which is 0xF6F6F6282828 in hexadecimal notation. Based on this sequence, 15 other code words are found, with the maximum possible pair-wise hamming distance. These are listed in Table 6.1. Like the OTN FAS, all code words are DC balanced (i.e. has equal number of zeros and ones) and have enough transitions to keep the clock recovery circuitry in the receiver in sync. The minimum hamming distance d between any two code words is 24. Hence, as long as the number of bit errors is below $d/2 = 12$, the receiver will still be able to obtain the correct codeword by maximum likelihood decoding[100]. The exact probability of decoding a wrong code word on a link with a certain BER can be calculated using Eq. 1, where k is the number of correctable errors, n is

TABLE 6.1

LIST OF FAS/CI CODE WORDS

Code word #	Code word (CI)	
0 (original)	F6 F6 F6	28 28 28
1 (discarded)	F9 F9 F9	14 14 14
2	CF CF CF	03 03 03
3	C0 C0 C0	3F 3F 3F
4	AC AC AC	4D 4D 4D
	A3 A3 A3	5A 5A 5A
6	9A 9A 9A	66 66 66
7	95 95 95	71 71 71
8	6A 6A 6A	8E 8E 8E
	65 65 65	99 99 99
10	5C 5C 5C	A5 A5 A5
11	53 53 53	B2 B2 B2
12	3F 3F 3F	C0 C0 C0
13	30 30 30	D7 D7 D7
14	09 09 09	EB EB EB
15	06 06 06	FC FC FC

The code identifier (CI) sequences are optimized for maximum pairwise hamming distance to increase error resilience, thus decreasing the probability of incorrect FEC decoding. To avoid false frame alignment this optimization is also done with respect to the FAS. For the same reason, codeword 1 is discarded, since a single bit shift brings it very close to code word 0.

the number of bits in the codeword and *P* is the bit error rate. For this specific case, *k*=11 and *n*=48.

$$1-\sum_{i=0}^{k}(1-P)^{(n-i)}\cdot P^{i}\cdot\binom{n}{i} \qquad (1)$$

Based on Eq. 1, the expected Codeword Error Rate (CER), i.e. the number of incorrectly decoded FAS/CI fields per received frame can be calculated. These calculations show that the error correction is sufficient to work at bit error rates as high as 10^{-2}, corresponding to a SNR of around 4.5 dB, while retaining a CER below 10^{-13}. However, the concatenated codes proposed here are able to function at an SNR approaching 0dB, at which point the CER will reach approximately 0.04 as depicted in Figure 6.3. Given the severe consequences of incorrect decoding, this level of protection is insufficient to ensure proper operation. Fortunately, since the outer FEC works on soft decision values, the CI error correction can be easily expanded to use soft decision bits instead of hard decision. The CER of the soft decision decoder will have an upper error bound of

$$P_B \leq \sum_{i=1}^{n} A_i Q\left(\sqrt{2i \cdot SNR}\right) \qquad (2)$$

where A is the weight distribution of the code, $Q()$ is the "Q-function" and i signifies the hamming distance for each element in A [100]. For the code used in this design, all codewords have the same weight and a minimum distance of 24. Hence, the equation simplifies to

$$P_B \leq A_{d\min} Q\left(\sqrt{2_{d\min} \cdot SNR}\right) \qquad (3).$$

As Figure 6.3 shows, using soft decision decoding will significantly improve the resilience of the CI field, allowing the CI to be decoded with an error rate below 10^{-10} at 0dB SNR. The cost of this extra resilience is additional hardware complexity in the CI decoder. To ensure the feasibility of using this decoder, a high performance proof of concept system has been implemented for an Altera Stratix IV FPGA (EP4S100G5F45I2). The hamming decoder takes up less than 1% of the available logic resources with a decoding latency of only 7 clock cycles at 77MHz. Since the decoding only needs to be performed once for each 16kB frame, an even smaller iterative circuit could be used at the cost of additional decoding latency.

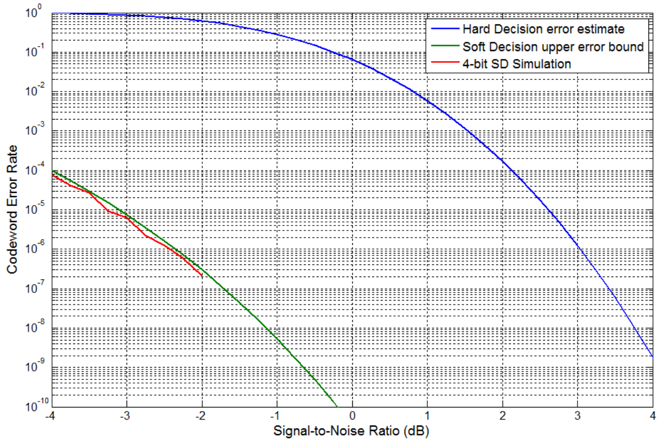

Figure 6.3 - Codeword Error Rate vs. SNR. The hard decision and the soft decision estimate curves are based on analytical calculations of the expected error rate. The 4-bit SD simulation shows the actual error correcting capabilities of the decoding algorithm simulated in Matlab using 4-bit soft decision resolution.

6.2.3.3. Frame Alignment

The purpose of the OTN Frame Alignment Sequence (FAS) is to allow the receiver to align itself to the boundaries of the incoming frame. Once aligned, the following boundaries are implicitly given by the fixed frame length and hence, for subsequent frames the FAS only serves to validate the correct alignment. In the proposed design, the length of a frame is no longer fixed but rather a function of the CI value. Furthermore, the system is designed to function at bit error rates (BER) above 10^{-2} which is much higher than standard OTN. Therefore, the OTN practice of scanning for error free FAS sequences to achieve and maintain frame lock, as depicted in Figure 6.4, is not suitable for this enhanced system. In order to locate the frame boundaries in this proposal, the following changes are made to the standard OTN frame aligner. The same modifications apply to detecting loss of frame alignment as well:

1) The FAS field is allowed to contain a small amount of errors *e* to facilitate alignment at higher bit error rates than standard OTN systems.
2) To reduce the risk of aligning to a false FAS (e.g. in the payload), the system scans for the whole 6 byte FAS as opposed to standard OTN, which only scans for the middle 4 bytes.
3) The correctness of the immediately following CI field serves to further verify the frame alignment.

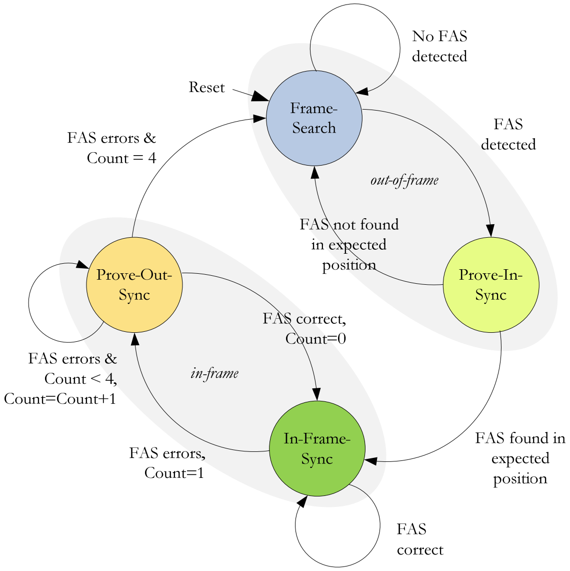

Figure 6.4 - OTN frame alignment state diagram [99]. In the proposed scheme, “FAS correct” implies number of FAS bit errors < *e*. Otherwise, it is perceived as an incorrect FAS.

4) The detected FAS/CI candidate now determines the distance to the next expected FAS/CI positions, as opposed to the fixed distance employed by OTN.

The number of allowed errors *e* in the FAS has a significant impact on how fast the system regains frame alignment after a service disruption has occurred. As depicted in Figure 6.4, two consecutive acceptable FAS sequences (errors <= *e*) are required to obtain a lock. If *e* is too low compared to the BER, the probability of receiving two such FAS sequences is correspondingly small. Hence, it will likely take several frames to obtain frame alignment. On the other hand, if *e* is too large, there will be a higher risk of false frame alignment, which is equally problematic. Due to the large FAS sequence used, and the fact that two sequences needs to be detected exactly one frame distance apart to obtain frame alignment, the issue of false frame alignment is only relevant for large values of *e*. To ascertain how the choice of *e* affects the frame alignment time of the system, the average number of frames to obtain alignment has been calculated as a function of the SNR for different values of *e*. The results are depicted in Figure 6.5. As seen, it is very important not to set *e* too low, as this may cause extremely long alignment times as well as a high risk of losing alignment during normal operation. For the very low SNR values, which are supported by the codes presented in this thesis, an *e* value of 5-8 is recommended for optimal performance.

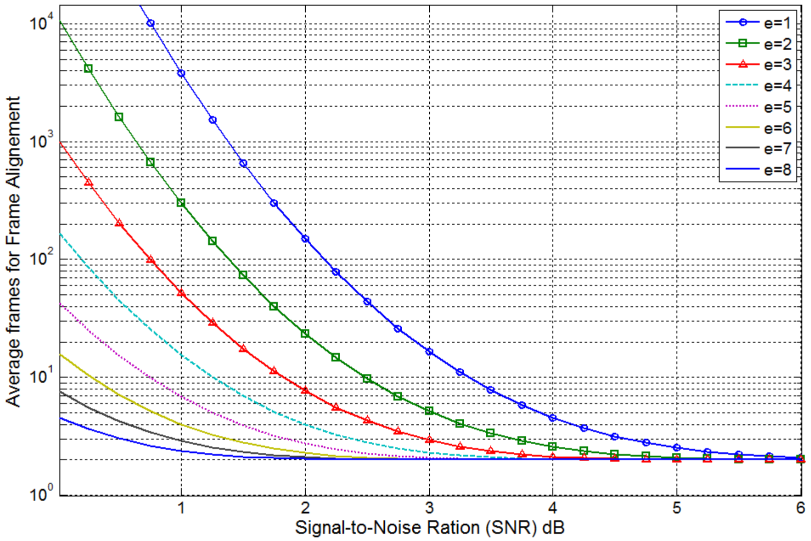

Figure 6.5 - Average number of frames to obtain FA for different values of acceptable FAS errors (e). Best case is 2 frames as defined by the standard OTN frame alignment state machine.

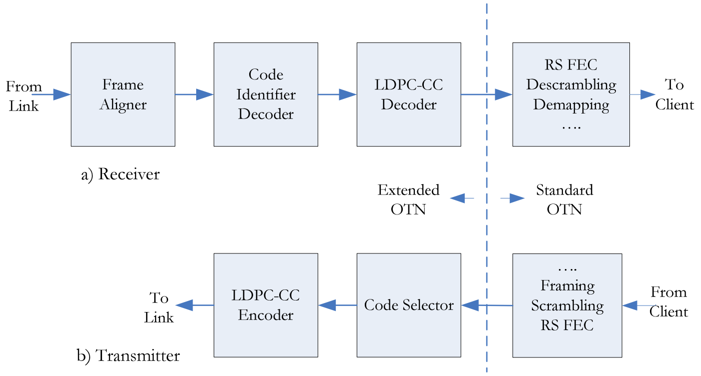

Figure 6.6 – Overall system schematic. Extra modules have been appended to a standard OTN system to allow for adaptive soft decision forward error correction (FEC).

The final system architecture is depicted in Figure 6.6. As seen, the adaptation circuit can be implemented as a simple add-on to the existing OTN system structure. On the transmitter side, the extension consists of a code selector, which chooses the codes for the individual frames based on the current operation parameters, and an adaptive LDPC-CC encoder. On the receiver side, the extension consists of an error tolerant Frame Aligner, a CI decoder and an adaptive LDPC-CC decoder.

6.2.4. Power Reduction Through Adaptive Iteration Reduction

As discussed in [101], it is possible to reduce the power consumption of an iterative FEC circuit by reducing the number of iterations performed on the data. This can be done if the SNR margin of the link is high enough to absorb the resulting loss in code gain. However, the SNR margin can also be artificially increased by selecting a FEC code with a lower rate, i.e. by sacrificing information throughput for additional parity bits. With more parity information to work with, the FEC decoder is able to achieve the same BER with less decoding iterations just as if the actual signal quality was increased. Hence, instead of operating a code of high rate with many iterations, it is possible to lower the rate and only use a couple of iterations to achieve the same BER. Since the receiver can see the code rate directly from the header of the incoming frames, the number of iterations performed by the iterative decoder could be configured accordingly based on predetermined BER estimates. However, to achieve maximum flexibility and performance, the proposal here is to let the FEC decoder itself determine the required number of iterations based on the estimated BER of its own output. The BER can be estimated for each bit based on its LLR value. It can be shown from the basic LLR equation in subsection 5.1.1, that the probability of an erroneous hard decision is given by

$$p_e = \frac{1}{1+\exp(|L|)}.$$

Hence, the average estimated BER at the output of the LDPC decoder after each iteration can be used as a stopping criterion for the iterative process. The goal, as mentioned in subsection 5.1.2, is to iterate just enough to get below the 10^{-4} BER limit, after which the RS code is able to handle the residual errors. Since the number of possible values of $|L|$ is usually quite small (e.g. 32 for a 6-bit LLR), the p_e values can be easily precomputed and stored in lookup tables to ease hardware implementation.

In order to save power in the decoder when running fewer iterations it is important that the actual decoder circuitry is designed with this feature in mind. The following section investigates how two different implementation strategies, using a serial or a parallel decoding approach, can be enhanced to support adaptive iteration reduction.

6.2.4.1. Iteration reduction in a serial decoder

In a serial iterative decoder, all iterations are performed by the same physical circuit. Hence, the required clock speed of the circuit is a function of the number of iterations and the desired decoder throughput. In order to convert this into reduced power consumption, the well known concept of Dynamic Frequency and Voltage Scaling (DFVS) can be used as suggested in [101] and described in [117]. The basic concept is to dynamically decrease the clock frequency driving the decoder to match the current load. Hence, if the maximum capacity of the iterative decoder is 8 iterations but only 2 iterations are required, the circuit can be driven at 1/4 of its maximum frequency. Since the power consumption of modern microchips increases approximately linearly with the clock frequency, this translates to a factor 4 reduction in the power consumption. In addition, when the clock frequency is reduced the circuit is able to run reliably at a lower core voltage, which reduces the power consumption even further. The approximate power consumption can be described by the equation $P = k \cdot F \cdot V^2$ [117][28], where k is a constant of the circuit, F is the clock frequency and V is the core voltage. A simulation of the normalized power consumption of the FEC as a function of the number of LDPC iterations is depicted in Figure 6.8. This figure is generated based on the same assumptions as described in [101]: The voltage range is $V_{min} \leq V \leq V_{max}$, where $V_{min} = 0.4 \cdot V_{max}$; V and F have the linear relationship $F = F_{max}/16$ at V_{min}. The RS decoder and the LDPC decoder are assumed to contribute with 1/10th and 9/10th of the total decoder power consumption, respectively, when the maximum number of LDPC iterations is used ($N = 8$). As seen from Figure 6.8, running the LDPC decoder at less than the maximum capacity significantly reduces the power consumption of the FEC.

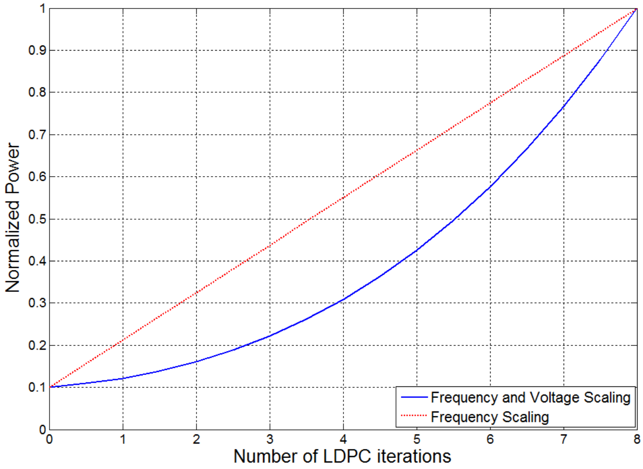

Figure 6.8 – Simulation of the normalized dynamic FEC power consumption. At zero LDPC iterations, only the hard decision RS FEC consumes power.

6.2.4.2. Iteration reduction in a parallel decoder

In order to enhance the throughput of the decoder, the iterations can be split into a pipeline structure of identical circuits, which are placed back-to-back. In this case, simply lowering the clock frequency and voltage is not a feasible solution as this will reduce the throughput of the FEC decoder correspondingly. Instead, the redundant iteration circuits can be bypassed once the estimated BER of a frame is low enough. The basic principle of such a system is depicted in Figure 6.7. Based on an estimation of the signal quality, the input from the link will run though N iteration elements to reduce the BER. Once the desired number of iterations has been reached, the processed data is then redirected through simple bypass elements instead of proceeding through the iteration chain. The bypass elements consist of a small multiplexer and a delay line and thus have minimal power consumption compared to the much more complex LDPC logic. By means of clock-gating [118], the remaining iteration circuits can be deactivated, thus eliminating the dynamic power dissipation and leaving only the much smaller static power dissipation (i.e. the leakage current)

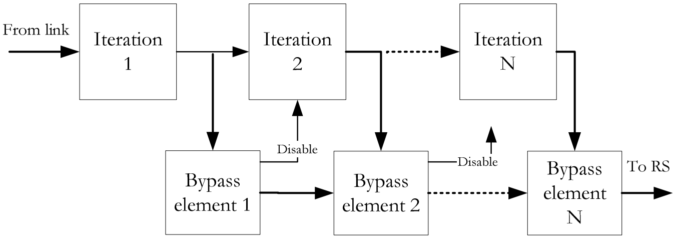

Figure 6.7 - Pipelined iterative FEC with dynamic iteration reduction (control signals not shown).

[28]. In this fashion, a reduction in the number of decoding iterations for a frame can provide an approximately linear reduction in the power dissipation of the LDPC decoder.

When calculating the power consumption of the FEC decoder at different code rates, the complexity in terms of number of computations per information bit (or in this case – per frame) of the LDPC decoder for each iteration also needs to be taken into account. For the family of LDPC codes investigated here, this complexity will be inversely proportional to the code rate when the information, i.e. the size of the inner OTN frame, is kept constant. Hence, even though the lower code rates allows for fewer decoding iterations, each iteration will consume more power. It is therefore important to balance these two contributions, when selecting the code rate. The codes analysed here are of rate R=b/(b+1), for b=1,2,…6. If the rate is reduced from R_{high} to R_{low}, the complexity will be increased by R_{high}/R_{low}. In the worst case, the rate will be reduced from $R_{high}=6/7$ to $R_{low}=1/2$, with the complexity increasing by 12/7≈2. In order to reduce the complexity overall, the number of iterations will have to be decreased sufficiently enough to compensate for the complexity increase due to the lower code rate. This term will be taken into account when estimating the overall power consumption reduction on a per-frame basis.

6.2.5. Simulation Results

This section presents simulation results, showing the different trade-offs in a flexible system employing LDPC-CC. The model are implemented using Matlab[119] and assumes an Additive White Gaussian Noise (AWGN) channel and Binary Phase Shift Keying (BPSK) modulation. However, the results are directly extendable to higher orders of modulation. As discussed in Section 5.1, the choice of the code memory M is crucial to performance, as well as to the hardware implementation complexity and scalability of the system. Preliminary studies using Matlab simulations have shown that $M = 257$ is a good compromise between performance and complexity. Higher memories, e.g. $M = 1025$ achieve an additional 0.5 dB gain, but since the goal is to estimate the relative power consumption reduction as a function of the code rate, $M = 257$ is chosen to reduce the simulation time. Very large memories, as mentioned in Section 5.1, are also impractical from a hardware implementation point of view, and $M = 257$ results are therefore also more relevant to real-life systems. The block length is $k = 240T = 61440$ bits.

As discussed in subsection 5.1.2, a BER of around 10^{-4} is needed at the input of the RS code for practically error-free transmission. In order to assess the necessary number of iterations, and thus the relative power consumption, 10000 blocks have been simulated for a number of code rates. Figure 6.9 reports the number of iterations after which a BER below the desired threshold is achieved for different code rates. The rate reduction due to termination is taken into account and the true rates as given in subsection 5.1.1.2 are shown in the legend. As can be seen, the number of iterations can be reduced by a factor of 2 to 4 by reducing the code rate, assuming a maximum iteration count of 8.

In Figure 6.10, the normalized power consumption is given per rate, taking into account the necessary number of iterations from Figure 6.9 and the power curve from Figure 6.8. As mentioned in Section 6.2.4, reducing the code rate increases the complexity in terms of the number of calculations needed per block. This factor is taken into account by multiplying the actual power consumption by 1/R for each code rate. As seen from Figure 6.10, the power consumption can be reduced by a factor of up to 4, by changing the code rate depending on the received SNR. For SNRs which allow successful decoding after just *one* iteration for all code rates, the complexity increase due to low code rate is dominating the power consumption. Hence, the power consumption per information bit in the high-SNR region (beginning at around 4dB) will actually *increase* for lower code rates. For SNR values above 8.5dB, the LDPC FEC can in fact be turned off all together, since the error rate of the link will already be low enough to be corrected by the outer RS code (see Figure 5.1, page 80). For lower SNR values, the simulation results show that the concatenated RS/LDPC code increases the code gain by up to 6.5dB (9dB on the physical link, before the FEC overhead is taken into account) compared to using just the RS code on its own.

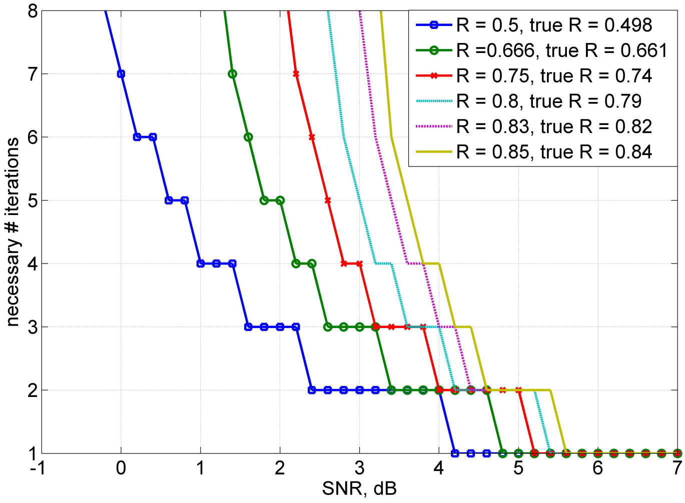

Figure 6.9 - Necessary number of iterations for achieving $BER \leq 10^{-4}$ for each code rate.

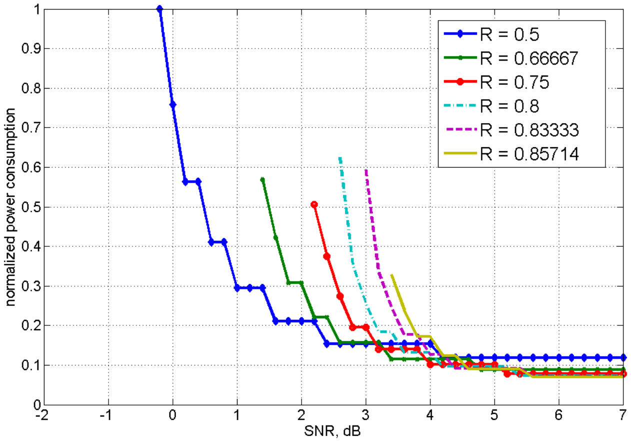

Figure 6.10 - Normalized power consumption, taking into account frequency and voltage reduction, and complexity increase due to lower rate, for each rate, achieving the desired threshold of $BER < 10^{-4}$.

Since the decoder will be designed to stop iterating based on the estimated reliability of the soft bits, it is also necessary to investigate the effects of BER estimation errors. As seen from Figure 6.10, operating the code with R=1/2 gives the least power consumption for the largest SNR region. Therefore, that code is chosen for the following analysis of the estimation errors. In Figure 6.11, the true (actual) BER is given, together with the estimated one. The results show that the estimated BER is in fact extremely close to the true BER. To investigate further, the worst case scenario is considered, where the decoder stops iterating, based on an underestimated BER. In Figure 6.12, the probability of an erroneous halt is given, together with the *effective BER estimation error,* for the $R = 1/2$ code after iterations 1 up to 5. The probability of an erroneous halt p_{err_halt} is calculated from the number of blocks, for which the estimated BER is below the 10^{-4} threshold, but the actual BER is not. If the average absolute estimation error is given as

$$E_{av} = \frac{\sum_{n=1}^{N} |true\_BER_n - est\_BER_n|}{N}, \qquad (4)$$

where N is the number of blocks, $true\_BER_n$ and est_BER_n are the actual BER and the estimated BER for each block, respectively, the effective BER estimation error is calculated as $p_{err\_halt} \cdot E_{av}$. Let us consider for example the case, where the decoder stops after the 2nd iteration. As seen from Figure 6.9, 2 iterations will usually be needed for an SNR between 2.4 and 4 dB if the code rate is 1/2. Figure 6.12 shows,

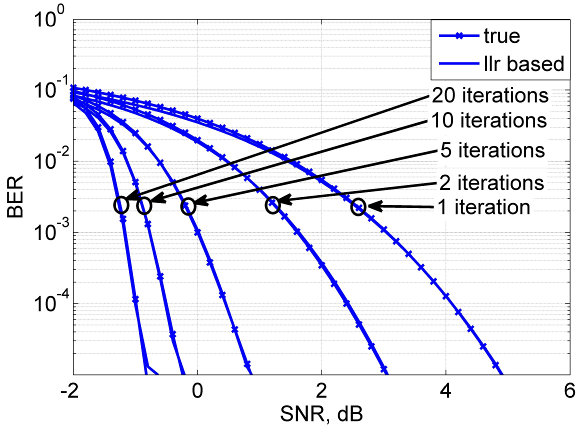

Figure 6.11 - True and estimated BER performance of a $R = \frac{1}{2}, M = 257,$ $k = 61440$ LDPC-CC with zero-padding termination.

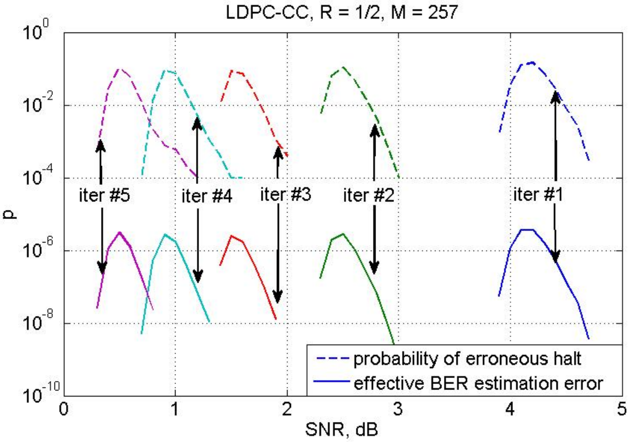

Figure 6.12 - Probability of erroneous halt and effective BER estimation error for the $R = 1/2, M = 257, k = 61440$ LDPC-CC.

that the BER estimation is perfect after 3 dB, and that the BER is underestimated for at most 10% of the blocks (dotted line) at around 2.5dB. The corresponding effective BER estimation error is around 10^{-6}, which for the desired BER threshold of 10^{-4} is negligibly small. This very precise estimation ensures that even if the LDPC decoder stops decoding prematurely, it will not affect the performance of the following RS code.

6.3. Chapter Summary

This chapter has provided an introduction to elastic optical transport networks and demonstrated a new scheme for energy conservation on optical links through adaptive forward error correction. The chapter demonstrates how a rate adaptive OTN-based transmission system can be built with minor adjustments to the standard OTN transceiver, while retaining backwards compatibility with existing equipment. The proposed system can perform rate adjustments on a frame-by-frame basis with no disruption to the traffic on the link. While adaptive FEC has several applications, this chapter has focused on using FEC adaptation of concatenated Reed-Solomon and soft decision LDPC-CC FECs as a means to reduce power consumption in the FEC decoder itself during periods of low load. The presented studies show that the number of necessary decoding iterations to reach error free transmission can be significantly reduced by lowering the code rate R at the cost of decreased throughput. By lowering the code rate, and thus the number of decoding iterations, during periods of low capacity demand the power consumption of the investigated FEC can be reduced by up to 75% compared to its nominal value. Since the scheme presented in this chapter can perform FEC adjustments on a frame-by-frame basis, it is able to provide a continuous power reduction, adapting both to the static conditions of the connection, as well as to the dynamics of the network on a microsecond timescale. As such, excess SNR margin and unused data bandwidth are readily converted into fewer joules per information bit with no ill effect to the live data traffic on the physical link.

7. Conclusion

The continuous growth in internet traffic is a constant driver for the development of higher link speeds and faster network nodes. As such, the demand for 40G and 100G links has been increasing over the course of this Ph.D. project, with new standards for 400G equipment under way. The work presented in this thesis aimed at aiding the evolution towards 100Gbps and beyond by investigating how core network node functionalities can be scaled up to support these high throughputs.

Since the start of the project "The Road to 100 Gigabit Ethernet", there have been significant improvements in the low-level hardware, which facilitates the development of equipment for 100G and beyond. This is especially true for the ASIC and the FPGA technology, but interesting advances have been made within the field of external memory as well. The standards themselves have also evolved, making 100G Carrier Ethernet a viable and potentially more flexible alternative to traditional connection oriented transport technologies. Even with these advances in the underlying technology, there are still plenty of challenges when moving beyond the 100G boundary from an architectural, algorithmic and digital design point of view.

One of the key components, which needs to scale linearly in performance with the link speed is the forwarding engine. In Chapter 3 of this thesis, this component has been investigated for both Ethernet switches (MAC tables) and IP routers (IP routing tables). The research within MAC tables has shown that the commonly used data structure known as a hash table is in fact able to scale to 100Gbps link speed using parallel memory access. The table design described in Chapter 3, using the Multilevel Adaptive Hash Table scheme, is thus able to process a frame in a single clock cycle, requiring a modest 150MHz clock frequency to processes 100GE traffic. This is done with the moderate memory requirement of 6.75Mbit of dual-port SRAM for a 64k MAC table. Depending on the achievable internal clock speed, one such table will be able to serve one or a few 100G ports. For high port counts, the scheme will still be able to scale well using distributed forwarding engines. For the IP routing tables, the methods used in Ethernet forwarding tables are not directly transferable. Instead, this thesis has proposed a novel LPM search engine structure, using a Ternary CAM (TCAM), which has been specifically tailored to perform LPM. The result is a TCAM with a much simpler and faster updating procedure and 100% memory utilization at the cost of around 16% extra logic compared to the standard TCAM. Trial implementations in an FPGA with 1,024 address entries have shown that this structure is capable of performing 411 million LPM lookups per second, corresponding to a forwarding rate of 411Mpps. If carried over standard Ethernet, this table would be able to handle a sustained line rate of approximately 275Gbps of using minimum sized IP packets. Hence, it is concluded that both the Ethernet and the IP forwarding engines can scale to 100G by means of these parallel search architectures, and also beyond 100G as the chip production technology improves.

Once the forwarding engine has determined the correct output port for a datagram, it needs to be moved across the switch fabric as efficiently as possible. The novel ERRMS multicast scheduling algorithm, presented in Chapter 4, uses a combination of queue look-ahead and synchronized output scheduling to get optimal multicast performance within the switch fabric. As opposed to its predecessor, the iterative MLRRMS scheme, it is optimized for parallel hardware and thus only needs one clock cycle to schedule cells for each cell time. This enables ERRMS to be used in a high-speed switching environment such as 100 Gigabit Ethernet switches. By taking the fan-out information of the first two multicast cells into account when scheduling, the ERRMS is able to increase the sustainable throughput significantly compared to the WBA while providing much better scalability compared to FIFOMS. Simulation results show that increasing the look-ahead depth beyond these two cells only provides marginal improvements to the throughput performance. The hardware complexity studies have shown that the ERRMS is capable of providing much faster multicast scheduling than the iterative MLRRMS for the implemented look-ahead depth of two cells, about a factor of two for all port configurations. The speedup is expected to grow in the same approximately linear fashion if the look-ahead depth is increased due to the parallel operation of ERRMS. Even though ERRMS performs scheduling on multiple levels in parallel, the logic size of the ERRMS scheduler has shown to be reasonable for hardware implementation and scales well to even very large switch fabrics. For a 32-port fabric, the scheduling performance of the FPGA implementation will be able to keep up with a 32Gbps traffic steam using a cell size of approximately 64B, given perfect cell utilization. This result indicates that an implementation based on modern ASIC technology or future FPGAs will be able to support the scheduling speeds required for 100G switching.

In Chapters 5 and 6 it has been investigated how the forward error correcting (FEC) codes, which are commonly employed in communication networks, can be enhanced to perform at higher speeds and provide increased bandwidth efficiency. The goal has been to provide better code gain, increased processing throughput and higher energy efficiency. Simulation results show that the standard Reed-Solomon (RS) (255,239) code can be concatenated with the more advanced Low Density Parity Check (LDPC) codes to increase the code gain by up to 6.5dB (9dB on the physical link, before the FEC overhead is taken into account), compared to Reed-Solomon on its own. Furthermore, it is shown how the RS and the LDPC codes can be implemented in parallel hardware to achieve 100Gbps and 10Gbps, respectively, on a modern FPGA. From these results, it is expected that an ASIC implementation will be able to contain a concatenated RS/LDPC circuit with more than 100Gbps of throughput.

To increase the energy efficiency of the physical links, a novel design for an energy effective, rate adaptive FEC system has been described in Chapter 6. The FEC is designed to be used both on its own, and in conjunction with other rate adaption parameters such as the symbol rate and the modulation format. The scheme opens up several opportunities for power conservation: Firstly, the adaptive FEC complements the overall adaptivity of the transmission system by allowing it to employ the most energy efficient combination of modulation format and FEC for a given capacity. Secondly, the internal processing inside the FEC decoder is fine-tuned based on the estimated output bit-error-rate (BER) to reach the BER goal with as little processing, and thus power consumption, as possible. Thirdly, by combining the properties of adaptive rate and adaptive processing effort, the transmitter is able to adjust the code rate of the individual frames based on the *exact* capacity demand on a microsecond time scale. Simulation results presented in this thesis have shown that reducing the code rate at the cost of lower effective throughput significantly reduces the amount of processing required in the FEC decoder for each frame. By employing simple techniques for performance/power tradeoff such as Dynamic Frequency and Voltage Scaling (DFVS), this reduced processing translates to an estimated reduction of the power consumption in the FEC decoder by up to 75%. Since the scheme presented in this thesis can perform FEC adjustments on a frame-by-frame basis, it is able to provide a constant power reduction, adapting both to the static conditions of the connection as well as to the dynamics of the network on a microsecond timescale. As such, excess SNR margin and unused data bandwidth are readily converted into fewer joules per information bit with no ill effect to the live data traffic on the physical link.

At the beginning of this Ph. D. project in 2010, 40G and 100G equipment was still very much in the development phase. Now, multiple vendors have put 100G equipment on the marked, and work is underway for drafting next generation 400G standards. The research done as part of this thesis indicates, that while the next step to 400Gbps still is a significant challenge, it is realistic to expect preliminary 400Gbps systems to emerge within the next 4-5 years. The basic technology which made 100Gbps possible is still evolving, with denser and faster integrated circuits (ICs), new high performance memory designs, such as the Hybrid Memory Cube (HMC), and increasing performance and availability of high speed serial links. Hence, advances in IC speed and density can be used to perform data processing faster and with even more parallelization. Combined with higher memory- and transceiver bandwidths to support 400Gbps of dataflow through the system, the techniques described in this thesis for 100Gbps will likely be able to scale to 400Gbps in the near future.

8. References

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2012–2017."

[2] IEEE, "Official Webpage of the IEEE P802.3ba 40Gb/s and 100Gb/s Ethernet Task Force." [Online]. Available: http://www.ieee802.org/3/ba/.

[3] Ixia, "Product Brief: Xcellon-Multis® 100/40/10GE Load Modules," 2013. [Online]. Available: http://www.ixiacom.com/products/xcellon/xcellon-multis/index.php.

[4] Xena Networks, "Xena Networks 100GE test module," 2013. [Online]. Available: http://www.xenanetworks.com/html/m1cfp100.html.

[5] Cisco, "Cisco Nexus 7700 F3-Series 12-Port 100 Gigabit Ethernet Module Data Sheet," 2013. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9402/data_sheet_c78-728423.html.

[6] IEEE, "400 Gb/s Ethernet Study Group," 2013. [Online]. Available: http://www.ieee802.org/3/400GSG/.

[7] C. Cole and I. Lyubomirsky, "400Gb / s is for Engineers and 1Tb / s is for Dreamers," in *Workshop 14 presented at ECOC 2011*, 2011, no. September.

[8] O. Gerstel and M. Jinno, "Elastic optical networking: A new dawn for the optical layer?," *IEEE Commun. Mag.*, no. February, pp. 12–20, 2012.

[9] LAN/MAN Standards Committee of the IEEE Computer Society, "IEEE standard 802.3ba-2010 Amendment: MAC parameters, physical layers and management parameters for 40 gb/s and 100 gb/s operation," .

[10] EO-Net Consortium, "EO-Net Official Webpage," 2013. [Online]. Available: http://www.celtic-

initiative.org/Projects/Celtic-projects/Call7/EO-Net/eonet-default.asp.

[11] M. Rowe, “40-Gbps and 100-Gbps Ethernet will bring new test challenges,” *Test & Measurement World*, 2009.

[12] Micron, “MT41J256M8 DDR3 SDRAM Datasheet.” 2006.

[13] S. Iyer, R. R. Kompella, and N. McKeown, “Designing Packet Buffers for Router Linecards,” *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 705–717, Jun. 2008.

[14] F. Wang and M. Hamdi, “Memory subsystems in high-end routers,” *IEEE Micro*, pp. 52–61, 2009.

[15] J. Garcia-Vidal, M. March, L. Cerda, J. Corbal, and M. Valero, “A DRAM/SRAM memory scheme for fast packet buffers,” *IEEE Trans. Comput.*, vol. 55, no. 5, 2006.

[16] Hybrid Memory Cube Consortium, “Official Webpage of the Hybrid Memory Cube Consortium,” 2013.

[17] MoSys, “Product Brief: MSR576 Bandwidth Engine® IC.” 2013.

[18] Rambus, “XDR™ Memory Architecture,” 2013. [Online]. Available: http://www.rambus.com/us/technology/solutions/xdr/.

[19] Hybrid Memory Cube Consortium, “Hybrid Memory Cube Specification 1 . 0.” 2013.

[20] M. J. Thomas Pawlowski, “Hybrid Memory Cube: Breakthrough DRAM Performance with a Fundamentally Re-Architected DRAM Subsystem,” in *Hot Chips 23th*, 2011.

[21] Cypress Semiconductor Coorporation, “QDR-II+ product information,” 2013. [Online]. Available: http://www.cypress.com/?id=108.

[22] M. Shreedhar and G. Varghese, “Efficient fair queueing using deficit round robin,” *ACM SIGCOMM Computer Communication Review*, vol. 25. pp. 231–242, 1995.

[23] R. Seifert and J. Edwards, *The All New Switch Book*, 2nd ed. Wiley Publishing, Inc., 2008.

[24] H. J. Chao, C. H. Lam, and E. Oki, *Broadband Packet Switching Technologies*, 1st ed. Wiley-Interscience, 2001.

[25] R. Rojas-Cessa, E. Oki, and H. J. Chao, “On the Combined Input-Crosspoint Buffered Switch With Round-Robin Arbitration,” *IEEE Trans. Commun.*, vol. 53, no. 11, pp. 1945–1951, Nov. 2005.

[26] R. Rojas-Cessa and Z. Dong, “Load-Balanced Combined Input-Crosspoint Buffered Packet Switches,” *IEEE Trans. Commun.*, vol. 59, no. 5, pp. 1421–1433, May 2011.

[27] S. Ruepp and A. Rytlig, “Performance evaluation of 100 Gigabit Ethernet switches under bursty traffic,” *15th Int. Conf. Opt. Netw. Des. Model. 2011*, 2011.

[28] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 4th ed. New York, New York, USA: Oxford University Press, 1998.

[29] Altera, “Altera FPGA Catalogue,” 2013. [Online]. Available: http://www.altera.com/devices/fpga/fpga-index.html.

[30] Xilinx, “Xilinx FPGA Catalogue,” 2013. [Online]. Available: http://www.xilinx.com/products/silicon-devices/fpga/index.htm.

[31] Intel, “Intel switch chip catalogue,” 2013. [Online]. Available: http://ark.intel.com/products/family/134/Intel-Ethernet-Switching-Components.

[32] IEEE Computer Society, “IEEE Std 802.1Q-2005: IEEE Standard for Local and metropolitan area networks Virtual Bridged Local Area Networks,” vol. 2005, no. May. 2006.

[33] IEEE Computer Society, *IEEE Std 802.1ad-2005: IEEE Standard for Virtual Bridged Local Area Networks Amendment 4□ : Provider Bridges*, vol. 2005, no. May. 2006.

[34] IEEE Computer Society, *IEEE Std 802.1ah-2008: IEEE Standard for Local and metropolitan area networks — Virtual Bridged Local Area*

Networks Amendment 7□ : Provider Backbone Bridges, vol. 2008, no. June. 2008.

[35] IEEE Computer Society, *Local and metropolitan area networks —*, vol. 2009, no. August. 2009.

[36] IEEE Computer Society, *IEEE Std 802.3ah-2004*, vol. 2004, no. September. 2004.

[37] IEEE Computer Society, *IEEE Std 802.1ag-2007: IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks—Amendment 5: Connectivity Fault Management*, vol. 2007. 2007.

[38] IEEE Computer Society, *IEEE 802.1aq-2012 - Shortest Path Bridging*, vol. 2012. 2012.

[39] EXFO, "EXFO product listing," 2010. [Online]. Available: http://www.exfo.com.

[40] Altera Corporation, "Altera product listing," 2013. [Online]. Available: http://www.altera.com.

[41] H. Veenstra and J. R. Long, *Circuit and interconnect design for rf and high bit-rate applications*. PrintPartners Ipskamp B.V., Enschede, The Netherlands., 2008.

[42] Altera Corporation, "Stratix IV Device Handbook." p. 15, 2009.

[43] E. Laskin and S. P. Voinigescu, "60 mw per lane, 4x23-gb/s 2(7)-1 prbs generator," *IEEE J. Solid-State Circuits*, vol. 41, no. 10, pp. 2198–2208, 2006.

[44] A. Kirsch, M. Mitzenmacher, and G. Varghese, "Hash-Based Techniques for High-Speed Packet Processing," pp. 1–40.

[45] A. Broder and A. Karlin, "Multilevel adaptive hashing," *Proc. first Annu. ACM-SIAM Symp. Discret. Algorithms*, 1990.

[46] A. I. Dumey, "Indexing for rapid random access memory systems," *Comput. Autom.*, vol. 5, no. 12, pp. 6–9, 1956.

[47] V. Papaefstathiou and I. Papaefstathiou, "A memory efficient, 100 Gb/sec MAC classification engine," *IEEE Conf. Local Comput. Networks 30th Anniv. (LCN'05)l*, p. 2 pp.–471, 2005.

[48] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood, "Fast hash table lookup using extended bloom filter: an aid to network processing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 181–192, 2005.

[49] R. Jain, "A comparison of hashing schemes for address lookup in computer networks," *IEEE Trans. Commun.*, vol. 40, no. 10, pp. 1570–1573, 1992.

[50] R. Pagh and F. F. Rodler, "Cuckoo hashing," *J. Algorithms*, vol. 51, no. 2, pp. 122–144, May 2004.

[51] C. Huntley, G. Antonova, and P. Guinand, "Effect of Hash Collisions on the Performance of LAN Switching Devices and Networks," in *Proceedings 2006 31st Conference on Local Computer Networks*, 2006, pp. 280–284.

[52] M. T. McClellan, J. Minker, and D. E. Knuth, "The Art of Computer Programming, Vol. 3: Sorting and Searching," *Mathematics of Computation*, vol. 28. p. 1175, 1974.

[53] A. Kirsch, M. Mitzenmacher, and G. Varghese, "Hash-Based Techniques for High-Speed Packet Processing," in in *Algorithms for Next Generation Networks*, G. Cormode and M. Thottan, Eds. London: Springer London, 2010, pp. 181–218.

[54] A. Z. Broder and A. R. Karlin, "System with a plurality of hash tables each using different adaptive hashing functions," 5,032,9871991.

[55] "AS6447 BGP Routing Table Analysis Report [online]," 2010. [Online]. Available: http://bgp.potaroo.net/as6447/.

[56] Z. Li, S. Zhang, and Y. Ma, "Decision Tree Based Algorithm for IPv6 Routing Table Lookup," *ICCT'06. Int. Conf. Commun. Technol.*, 2006.

[57] Y. Fu, H. Liang, and Z. Liu, "An effective IP routing lookup algorithm based on network processor," *2008 11th IEEE Singapore Int. Conf. Commun. Syst.*, pp. 1716–1720, Nov. 2008.

[58] V. Srinivasan and G. Varghese, "Fast address lookups using controlled prefix expansion," *ACM Trans. Comput. Syst.*, vol. 17, no. 1, pp. 1–40, 1999.

[59] C. S. Inc., "Cisco Catalyst 6500 Architecture." 2007.

[60] G. Liao, H. Yu, and L. Bhuyan, "A new IP lookup cache for high performance IP routers," *Proc. 47th Des. Autom. Conf. - DAC '10*, p. 338, 2010.

[61] H. Lee, S. Ha, Y. Choi, and C. Science, "Fast Update of Forwarding Tables in Internet Router Using AS Numbers," pp. 341–346, 2001.

[62] A. J. McAuley and P. Francis, *Fast routing table lookup using CAMs.* IEEE Comnput. Soc. Press, 1993, pp. 1382–1391.

[63] Y.-S. C. Y.-S. Chu, H.-K. S. H.-K. Su, P.-F. L. P.-F. Lin, and M.-J. C. M.-J. Chen, "High speed routing lookup IC design for IPv6," in *2006 IEEE International Symposium on Circuits and Systems*, 2006.

[64] W. W. W. Wu, B. S. B. Shi, and F. W. F. Wang, "Fast updating scheme of forwarding tables on TCAM," in *The IEEE Computer Societys 12th Annual International Symposium on Modeling Analysis and Simulation of Computer and Telecommunications Systems 2004 MASCOTS 2004 Proceedings*, 2004.

[65] W. W. W. Wu and R. W. R. Wang, "A TCAM Management Scheme for IP Lookups," in *2006 14th IEEE International Conference on Networks*, 2006, vol. 1.

[66] M. J. Akhbarizadeh, M. Nourani, and C. D. Cantrell, *Segregating the encompassing prefixes to enhance the performance of packet forwarding engines*, vol. 3. 2004, pp. 1612 – 1616 Vol.3.

[67] E. Ng and G. L. G. Lee, "Eliminating sorting in IP lookup devices using partitioned table," in *2005 IEEE International Conference on ApplicationSpecific Systems Architecture Processors ASAP05*, 2005, pp. 119–126.

[68] R. W. Baldwin and E. Ng, "Technique to eliminate sorting in IP packet forwarding devices," in *IEEE International Conference on*

Computer Design VLSI in Computers and Processors 2004 ICCD 2004 Proceedings, 2004, pp. 554–559.

[69] V. Ravikumar and R. Mahapatra, "TCAM architecture for IP lookup using prefix properties," *Micro, IEEE*, pp. 60–69, 2004.

[70] S. K. Maurya and L. T. Clark, "Low power fast and dense longest prefix match content addressable memory for IP routers," *Proc. 14th ACM/IEEE Int. Symp. Low power Electron. Des. - ISLPED '09*, p. 389, 2009.

[71] S. Maurya and L. Clark, "A Dynamic Longest Prefix Matching Content Addressable Memory for IP Routing," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 6, pp. 963–972, 2011.

[72] S. Kniesburges and C. Scheideler, "Hashed Patricia Trie: Efficient longest prefix matching in peer-to-peer systems," *LNCS*, vol. 6552, pp. 170–181, 2011.

[73] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet routing instability," *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, pp. 515–528, 1998.

[74] M. Kobayashi, T. Murase, and a. Kuriyama, "A longest prefix match search engine for multi-gigabit IP processing," *2000 IEEE Int. Conf. Commun. ICC 2000. Glob. Converg. Through Commun. Conf. Rec.*, vol. 3, pp. 1360–1364, 2000.

[75] B. Gamache, "A Fast Ternary CAM Design for IP Networking Applications," in *ICCCN*, 2003, vol. 00, no. C, pp. 434–439.

[76] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE J. Circuits*, vol. 41, no. 3, pp. 712–727, 2006.

[77] M. Karol, M. Hluchyj, and S. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," *IEEE Trans. Commun.*, vol. 35, no. 12, pp. 1347–1356, 1987.

[78] S. C. Liew, "Performance of various input-buffered and output-buffered ATM switch design principles under bursty traffic: simulation study," *IEEE Trans. Commun.*, vol. 42, 1990.

[79] E. Oki, R. Rojas-cessa, and H. J. Chao, “A Pipeline-Based Approach for Maximal-Sized Matching Scheduling in Input-Buffered Switches,” *IEEE Commun. Lett.*, vol. 5, no. 6, pp. 263–265, 2001.

[80] A. Smiljanic, “RRGS-round-robin greedy scheduling for electronic/optical terabit switches,” in *GLOBECOM’99*, 1999, pp. 1244–1250.

[81] L. Liu, Z. Zhang, and Y. Yang, “Packet Scheduling in a Low-Latency Optical Interconnect With Electronic Buffers,” *J. Light. Technol.*, vol. 30, no. 12, pp. 1869–1881, Jun. 2012.

[82] N. McKeown and B. Prabhakar, “Scheduling multicast cells in an input-queued switch,” in *Proceedings of IEEE INFOCOM 96 Conference on Computer Communications*, 1996, vol. 1.

[83] B. Prabhakar, N. McKeown, and R. Ahuja, “Multicast scheduling for input-queued switches,” *IEEE J. Sel. Areas Commun.*, vol. 15, no. 5, pp. 855–866, 1997.

[84] A. Bianco, P. Giaccone, E. Leonardi, F. Neri, and C. Piglione, “On the number of input queues to efficiently support multicast traffic in input queued switches,” in *Workshop on High Performance Switching and Routing 2003 HPSR*, 2003, pp. 111–116.

[85] A. Bianco, P. Giaccone, C. Piglione, and S. Sessa, “Practical algorithms for multicast support in input queued switches,” in *2006 Workshop on High Performance Switching and Routing*, 2006, pp. 187–192.

[86] M. Shoaib, “Selectively Weighted Multicast Scheduling Designs For Input-Queued Switches,” in *2007 IEEE International Symposium on Signal Processing and Information Technology*, 2007.

[87] H. Y. H. Yu, “A Novel Round-Robin Based Multicast Scheduling Algorithm for 100 Gigabit Ethernet Switches,” in *INFOCOM IEEE Conference on Computer Communications Workshops 2010*, 2010, pp. 1–2.

[88] B. Hu, C. He, and K. L. Yeung, “Achieving 100% Throughput for Multicast Traffic in Input-Queued Switches,” in *2011 IEEE Global Telecommunications Conference GLOBECOM 2011*, 2011, no. 2010, pp. 1–5.

[89] H. Yu, S. Ruepp, and M. S. Berger, "Enhanced first-in-first-out-based round-robin multicast scheduling algorithm for input-queued switches," *IET Commun.*, vol. 5, no. 8, p. 1163, 2011.

[90] H. Yu, S. Ruepp, and M. S. Berger, "Multi-Level Round-Robin Multicast Scheduling with Look-Ahead Mechanism," in *2011 IEEE International Conference on Communications ICC*, 2011, pp. 1–5.

[91] D. Pan and Y. Yang, "FIFO-based multicast scheduling algorithm for virtual output queued packet switches," *IEEE Trans. Comput.*, vol. 54, no. 10, pp. 1283–1297, 2005.

[92] D. P. D. Pan and Y. Y. Y. Yang, "Bandwidth guaranteed multicast scheduling for virtual output queued packet switches," in *2nd International Conference on Broadband Networks 2005*, 2005, vol. 1, no. d, pp. 981–990.

[93] E. S. Shin, V. J. Mooney, and G. F. Riley, "Round-robin arbiter design and generation," *Proc. 15th Int. Symp. Syst. Synth. ISSS 02*, p. 243, 2002.

[94] P. Gupta and N. McKeown, "Designing and implementing a fast crossbar scheduler," *IEEE Micro*, vol. 19, no. 1, pp. 20–28, 1999.

[95] Y. L. Lee, J. M. Jou, Y. Y. Chen, and S. S. Wu, "A Optimal Arbiter Design for NoC," in *International Computer Symposium ICS08*, 2008, pp. 288–293.

[96] H. Fatih Ugurdag and O. Baskirt, "Fast parallel prefix logic circuits for n2n round-robin arbitration," *Microelectronics J.*, vol. 43, no. 8, pp. 573–581, Aug. 2012.

[97] J. F. Hayes, R. Breault, and M. K. Mehmet-Ali, "Performance analysis of a multicast switch," *IEEE Trans. Commun.*, vol. 40, no. 10, pp. 581–587, 1991.

[98] "OPNET Modeler." [Online]. Available: http://www.opnet.com.

[99] ITU-T, "Recommendation G.709/Y.1331 (02/01), Interfaces for the optical transport network (OTN)." Geneva, 2001.

[100] S. Lin and D. Costello, *Error Control Coding*, Second edi. Pearson Prentice Hall, 2004.

[101] C. Dorize, O. Rival, and C. Costantini, "Power scaling of LDPC decoder stage in long haul networks," in *Photonics in Switching*, 2012, pp. 2–4.

[102] N. Axvig, D. Dreher, K. Morrison, E. Psota, L. C. Perez, and J. L. Walker, "Average min-sum decoding of LDPC codes," *2008 5th Int. Symp. Turbo Codes Relat. Top.*, pp. 356–361, Sep. 2008.

[103] A. Jimenez Felstrom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2181–2191, 1999.

[104] A. E. Pusane, A. Jim, A. Sridharan, M. Lentmaier, K. S. Zigangirov, and D. J. Costello, "Implementation Aspects of LDPC Convolutional Codes," *IEEE Trans. Commun.*, vol. 56, no. 7, pp. 1060–1069, 2008.

[105] Y. Miyata, W. Matsumoto, H. Yoshida, T. Mizuochi, and P. Ber, "Efficient FEC for Optical Communications using Concatenated Codes to Combat Error-floor," in *OFC/NFOEC*, 2008, pp. 11–13.

[106] I. T. U. (ITU), "ITU Recommendation G.709/Y.1331: Interface for the optical transport network (OTN)." pp. 34, 35, 108, 109, 2003.

[107] P. Layec, C. Dorize, O. Rival, F. Gioulekas, M. Birbas, C. Petrou, A. Vgenis, A. Rasmussen, and M. Karlsson, "D1.2: Report on modulation schemes and FEC performances and implementation for elastic WDM transmission systems.," *Deliv. Celt. Proj. CP7-006, EO-Net - WP1.*, 2012.

[108] Z. Chen, T. Brandon, and D. Elliott, "Jointly designed architecture-aware LDPC convolutional codes and high-throughput parallel encoders/decoders," *IEEE Trans. Circuits Syst.*, vol. 57, no. 4, pp. 836–849, 2010.

[109] A. F. Molisch, *Wireless Communications*. John Wiley & Sons Ltd, 2005.

[110] M. Jinno and H. Takara, "Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling

technologies," *IEEE Commun. Mag.*, no. November, pp. 66–73, 2009.

[111] G.-H. Gho, L. Klak, and J. M. Kahn, "Rate-Adaptive Coding for Optical Fiber Transmission Systems," *J. Light. Technol.*, vol. 29, no. 2, pp. 222–233, 2011.

[112] G.-H. Gho and J. M. Kahn, "Rate-Adaptive Modulation and Low-Density Parity-Check Coding for Optical Fiber Transmission Systems," *J. Opt. Commun. Netw.*, vol. 4, no. 10, pp. 760–768, Sep. 2012.

[113] J. G. Proakis and M. Salehi, *Communication Systems Engineering*. Prentice Hall, 1994.

[114] F. Vacondio, O. Rival, Y. Pointurier, C. Simonneau, L. Lorcy, J. Antona, and S. Bigo, "Coherent Receiver Enabling Data Rate Adaptive Optical Packet Networks," in *ECOC Technical Digest*, 2011, pp. 6–8.

[115] F. Vacondio, C. Simonneau, A. Voicila, E. Dutisseuil, and J. Tanguy, "Real time implementation of packet-by-packet polarization demultiplexing in a 28 Gb/s burst mode coherent receiver," in *Optical Fiber Communication Conference*, 2012, vol. 1, no. d, pp. 57–59.

[116] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[117] D. Ma and R. Bondade, "Enabling Power-Efficient DVFS Operations on Silicon," *Circuits Syst. Mag. IEEE*, pp. 14 – 30, 2010.

[118] J. F. Wakerly, *Digital Design: Principles and Practices*, 3rd ed. New Jersey: Prentice Hall, 2001.

[119] MathWorks, "Official Webpage of Matlab," 2013. [Online]. Available: http://www.mathworks.se/products/matlab/.

[120] Micron, "Micron's Official Webpage." [Online]. Available: http://www.micron.com.

ISBN: 978-87-93089-30-3

Published by:
DTU Fotonik
Department of Photonic Engineering
Technical University of Denmark
Ørsteds Plads, building 343
DK-2800 Kgs. Lyngby

Anders Rasmussen received the M.Sc.E.E. degree within the field of telecommunication from the Technical University of Denmark in 2009. After completing the degree, Anders Rasmussen worked as a research assistant in the Network Technology and Service Platforms group at DTU Fotonik and in 2010 he began to pursuit his Ph.D. studies on energy efficient high speed telecommunication systems.

During his Ph. D. studies Anders Rasmussen took part in the Danish national project “The Road to 100 Gigabit Ethernet” and the European CELTIC project “Elastic Optical Networks (EO-Net).