TCP/IP in hardware using SME

Mark Jan Jacobi & Jan Meznik

KU

September 18, 2019

2019-09-18

TCP/IP in hardware using SME

Mark siger introduktion og 2-3 saetninger "abstrakt"

Table of Contents

Table of Contents
Introduction
Implementation
Evaluation
Discussion
Conclusion
Future Work
Questions

Background and Motivation

FPGAs are making their way into data centers to boost the computing power and the overall power efficiency.

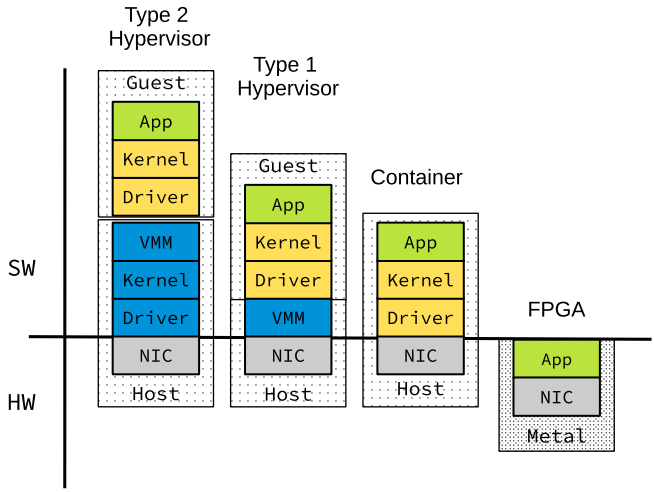
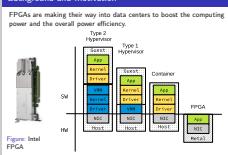


Figure: Intel FPGA



Applikationer og Big-Data udregninger flytter til Cloud, drevet af store data centre.

data-centre kraever meget plads, store maengder af stroem og er svaere at vedligeholde og udvide.

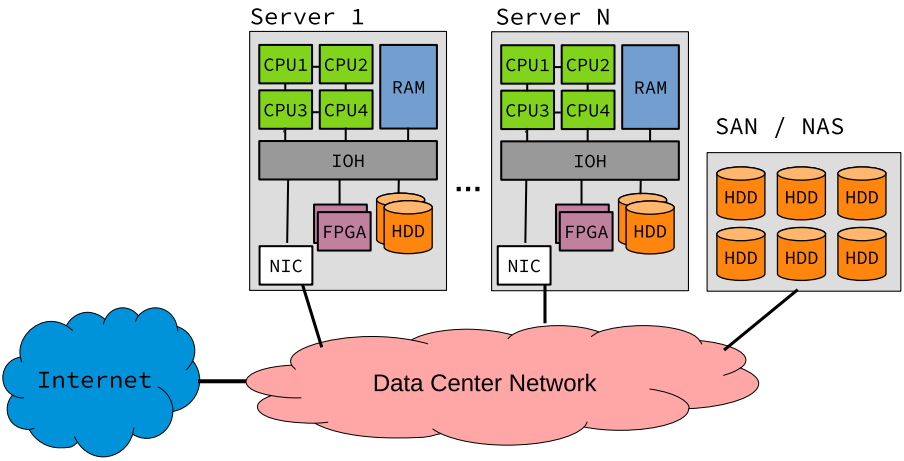
DC optimerer servere for at få mest værdi muligt

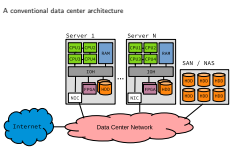
Tendens til aflaste beregninger til FPGAer, fjerne overhead

FPGA er hardware kan udføre beregninger hurtigt pga. dens parallele programmerbare natur. Den er hurtigt fordi instruktioner skrives direkte ned i hardwaren

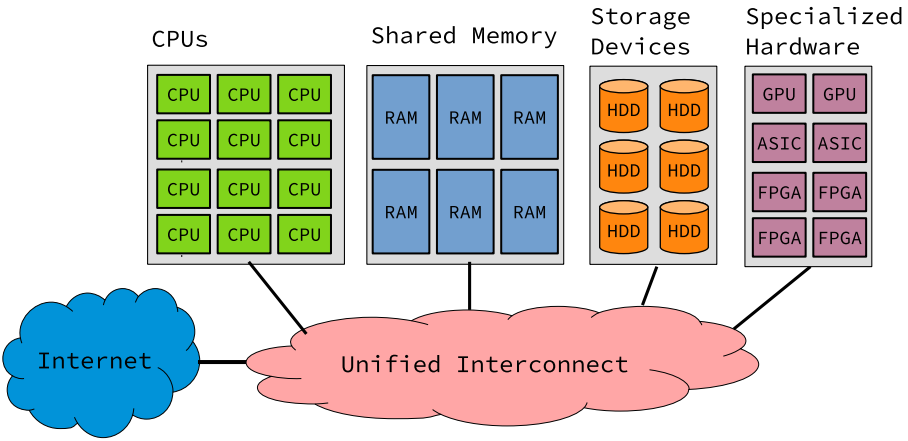
— GRAF HER —
PROBLEMET er at der kun kan vaere en begreanset antal af FPGAer i

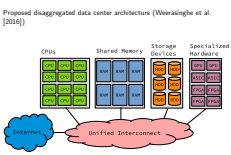
A conventional data center architecture





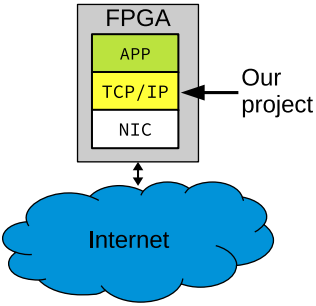
Proposed disaggregated data center architecture (Weerasinghe et al. [2016])





Hvis man splitter resourcerne op, kan man takket været FPGA få bedre ydeevne på det samme areal, samt nemmere håndtering af servere og deres komponenter.

FPGA usage



FPGA usage

The Internet

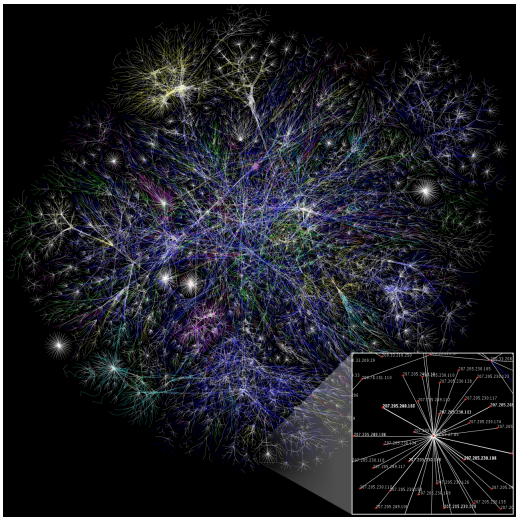


Figure: Map of about 30% of the accessible the endpoints on the Internet

The Internet

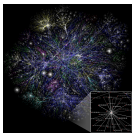
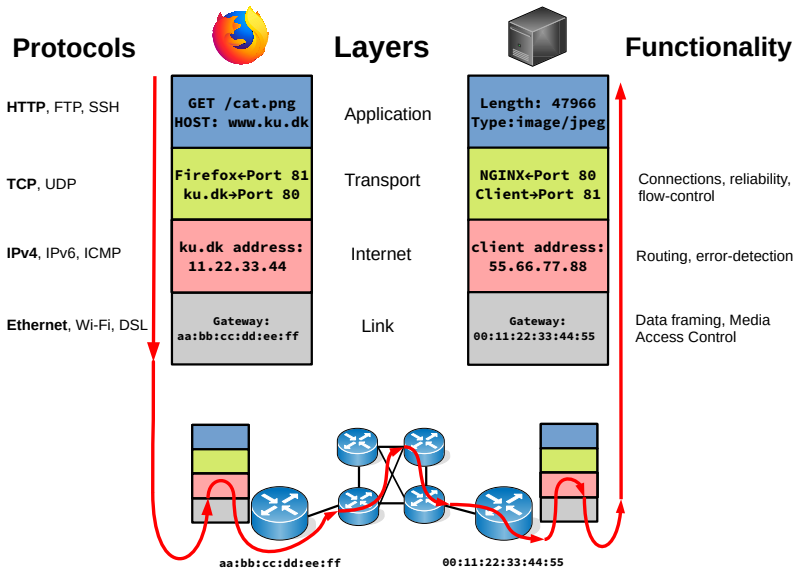
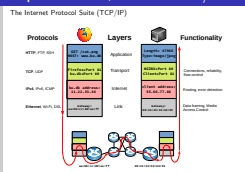


Figure: Map of about 30% of the accessible the endpoints on the Internet

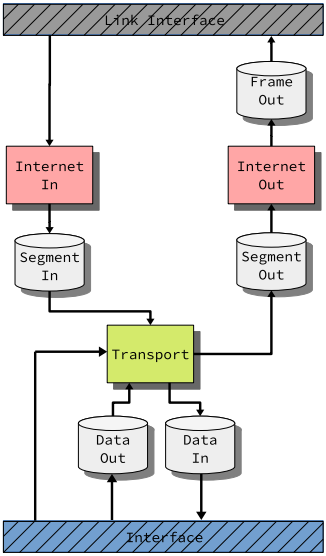




TCP/IP er samling af standarder og protokoller

Link: Overførsel på det fysiske medium
Internet: bestemmer data-veje, addressering, fejl-kontrol
Transport: pålidelighed, forbindelser, kontrol flow
Application: Defineret af selve applikationen

Design with the 4 layers in mind



Design with the 4 layers in mind

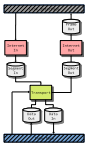


Table of Contents

Table of Contents

Introduction
Implementation
Evaluation
Discussion
Conclusion
Future Work
Questions

Implementation

- SME introduction
- Processes
  - State machines
- Buffers
  - Memory segments
  - Dictionary
- Interface signal control
  - Buffer-Producer
  - Compute-Producer

Implementation

- SME introduction
- Processes
  - State machines
- Buffers
  - Memory segments
  - Dictionary
- Interface signal control
  - Buffer-Producer
  - Compute-Producer

SME(Synchronous Message Exchange) introduction

- Processes and Busses
- Higher abstraction
- Handling of clocks
- Easy testing
- Not fully feature complete with C#(No threads, no allocation)

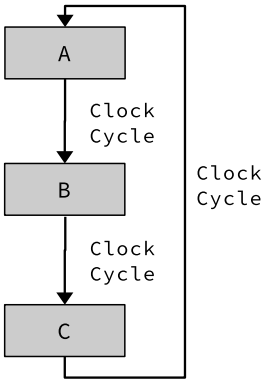
SME(Synchronous Message Exchange) introduction
- Processes and Busses
- Higher abstraction
- Handling of clocks
- Easy testing
- Not fully feature complete with C#(No threads, no allocation)

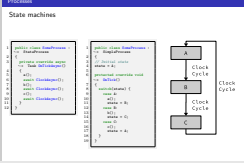
- What is a bus and a process
- No VHDL code
- Clocks abstracted away behind the management of processes and busses
- Testing straight in the simulator, but also in afterwards in the GHDL compiler, via an clock lookup table

State machines

```
1  public class SomeProcess :
   ↪ StateProcess
2  {
3    private override async
     ↪ Task OnTickAsync()
4    {
5      a();
6      await ClockAsync();
7      b();
8      await ClockAsync();
9      c();
10     await ClockAsync();
11   }
12 }
```

```
1  public class SomeProcess :
   ↪ SimpleProcess
2  {
3  // Initial state
4  state = A;
5
6  protected override void
   ↪ OnTick()
7  {
8    switch(state) {
9      case A:
10       a();
11       state = B;
12     case B:
13       b();
14       state = C;
15     case C:
16       c();
17       state = A;
18   }
19 }
```

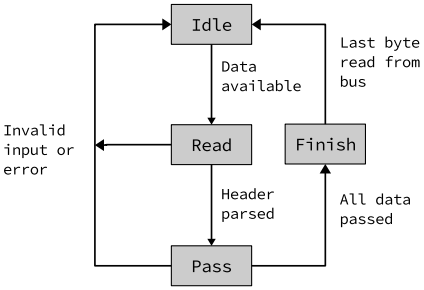




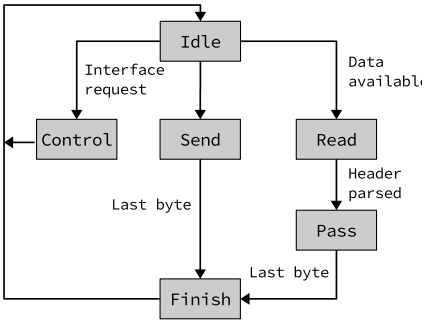
State machines

- `StateProcess`
  Eksekvering kan stoppes når som helst(i bidder)
- `SimpleProcess`
  Run er en clock altid, state machine håndteres med en switchcase.
  Algoritme kan splittes op i flere bidder, men kræver en state per bid

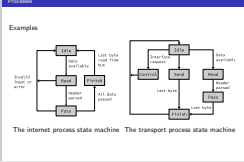
Examples



The internet process state machine



The transport process state machine



Labels!

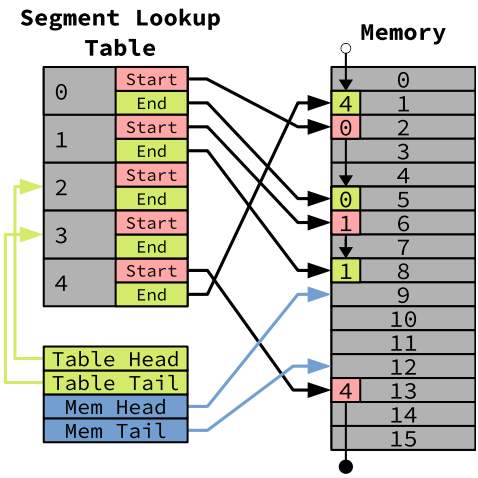
Implementation
Buffers

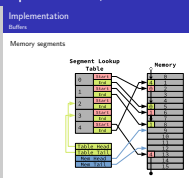
- why buffes?

Implementation
Buffers
- why buffes?

Hvorfor bruer vi buffers?

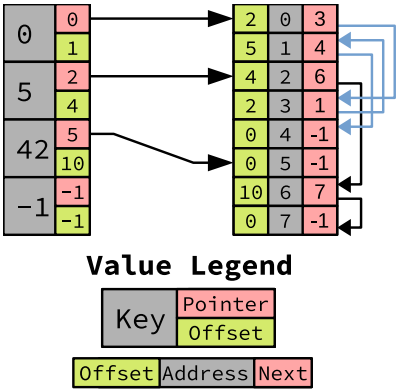
Memory segments

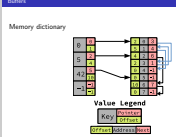




Thumbnail?

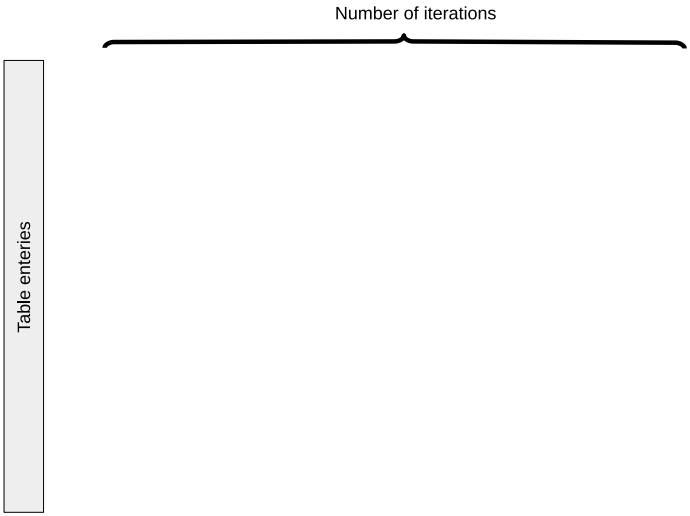
Memory dictionary





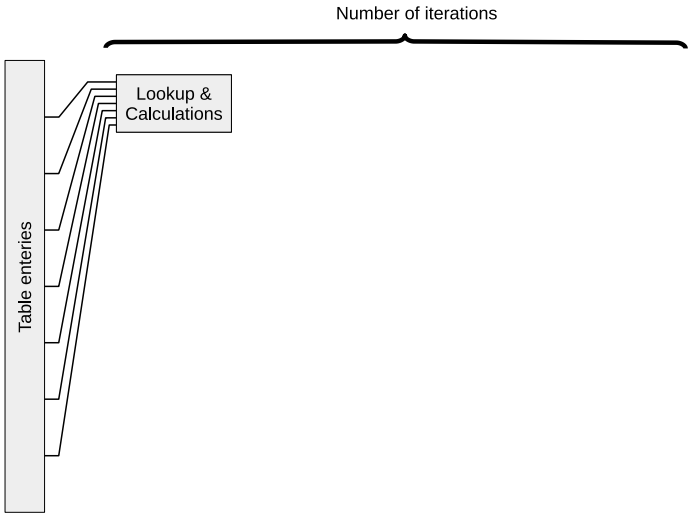
Statisk tabel?
Animationer? Billede af splitup af pakker? Hold det uden protocol specifik info.

Some problems with the memory dictionaries!

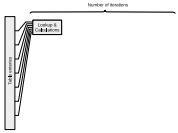


Implementation
Buffers
Some problems with the memory dictionaries!

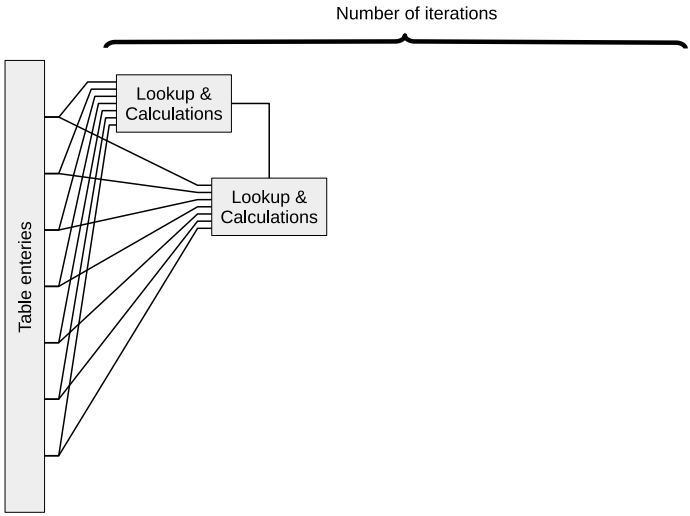
Some problems with the memory dictionaries!

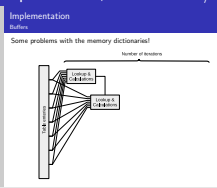


Implementation
Buffers
Some problems with the memory dictionaries!

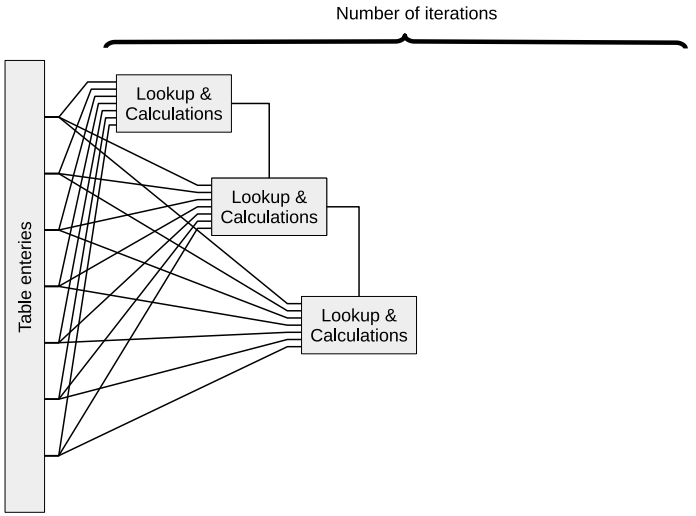


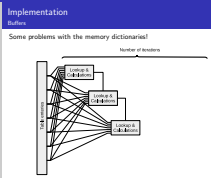
Some problems with the memory dictionaries!



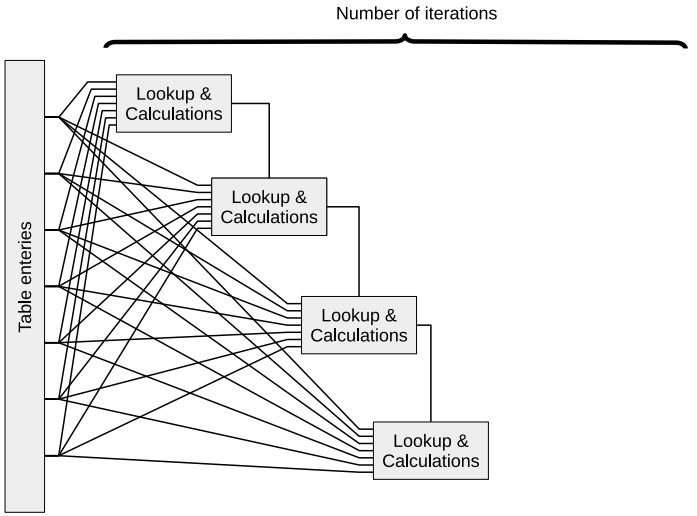


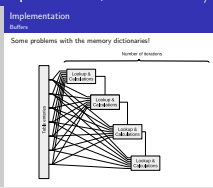
Some problems with the memory dictionaries!



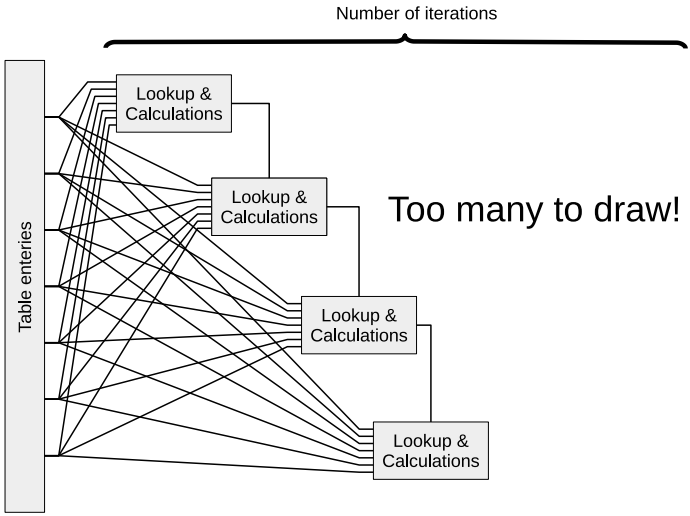


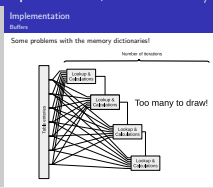
Some problems with the memory dictionaries!



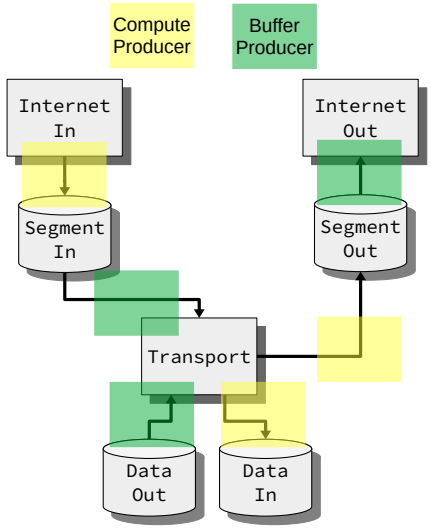


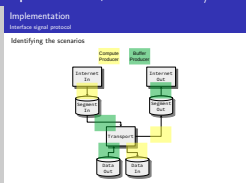
Some problems with the memory dictionaries!





Identifying the scenarios

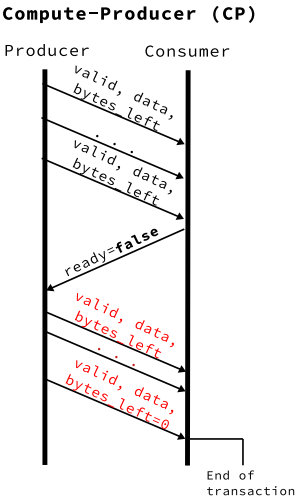




Data skal overføres hurtigst muligt, og det må ikke gå tabt

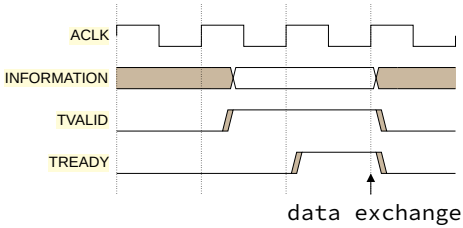
2 scenarier: fra "compute" til buffer, og omvendt

- CP kan ikke vente
- BP har stor buffer, og consumer starter transaktion

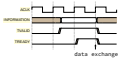


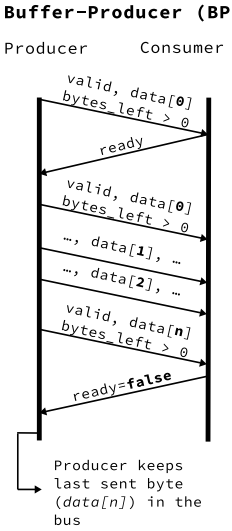
Implementation
Interface signal protocol

Buffer-Producer: Inspired by AXI4



Implementation
Interface signal protocol
Buffer-Producer: Inspired by AXI4





Implementation
Interface signal protocol

Table of Contents

Table of Contents

Introduction

Implementation

Evaluation

Discussion

Conclusion

Future Work

Questions

Evaluation

- Setup
  - Graph file simulator
- Test
- Validation
  - Latency
  - Outgoing packet validation
  - Internet Protocol Suite compliancy as per RFC 1122

Evaluation
- Setup
  - Graph file simulator
- Test
- Validation
  - Latency
  - Outgoing packet validation
  - Internet Protocol Suite compliancy as per RFC 1122

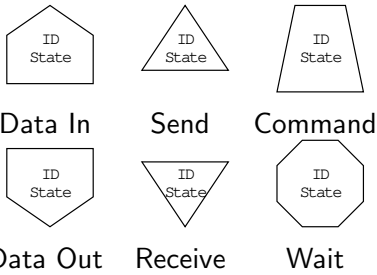
Graph file simulation

- Full input - output
- Does not take latency between packets into account
- Simplifies test cases

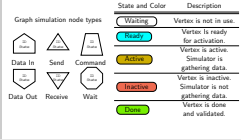
Evaluation
Setup
Graph file simulation
- Full input - output
- Does not take latency between packets into account
- Simplifies test cases

Definer send og receive bedre

Graph simulation node types

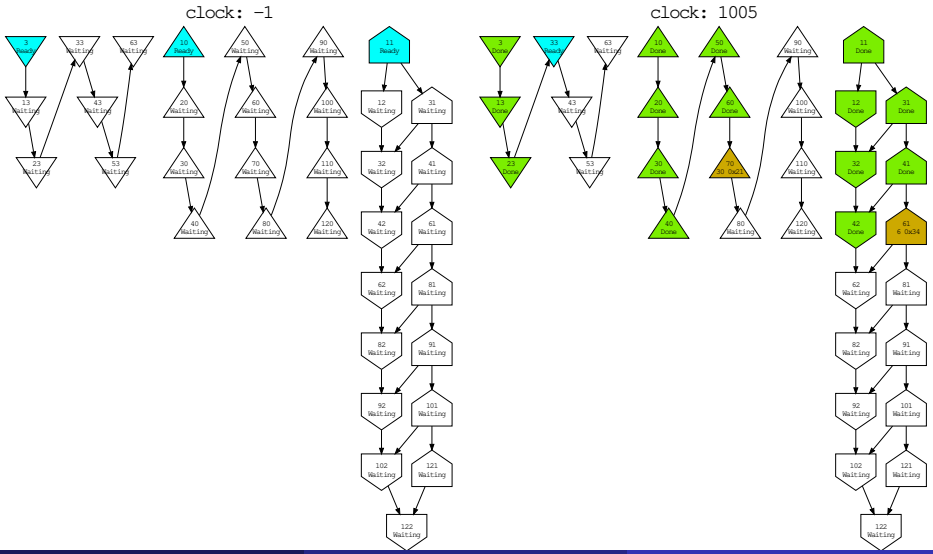


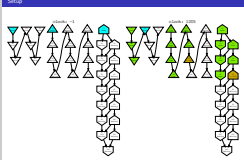
State and Color	Description
Waiting	Vertex is not in use.
Ready	Vertex Is ready for activation.
Active	Vertex is active. Simulator is gathering data.
Inactive	Vertex is inactive. Simulator is not gathering data.
Done	Vertex is done and validated.



Evaluation

Setup





Senario

- Real life scenario
- Test at high workloads
- Remove garbage
- Respond to packet
- Differ between concurrent connections

Evaluation
Test
Senario
Real life scenario
Test at high workloads
Remove garbage
Respond to packet
Differ between concurrent connections

The test

- 17283 packets in total
- Two "sessions"
- 640*2 UDP packets that needs a response
- 640 well formed UDP packets with no session (discard)
- Rest of data is "background noise" (TCP packets with state, data, etc)
- Total data sent through: 1832958 bytes
- 1.83 Million clocks used

Evaluation
Test
The test
17283 packets in total
Two "sessions"
640*2 UDP packets that needs a response
640 well formed UDP packets with no session (discard)
Rest of data is "background noise" (TCP packets with state, data, etc)
Total data sent through: 1832958 bytes
1.83 Million clocks used

Latency calculations:

n_D : The number of bytes in the data part of the protocol. This excludes both headers from transport and internet.

n_I : The internet header size.

n_T : The transport header size.

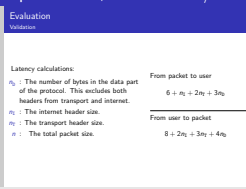
n : The total packet size.

From packet to user

$$6 + n_I + 2n_T + 3n_D$$

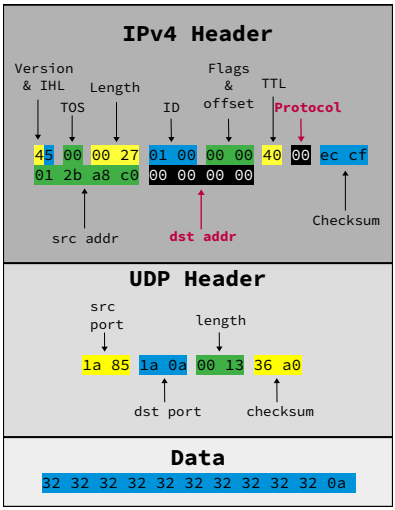
From user to packet

$$8 + 2n_I + 3n_T + 4n_D$$

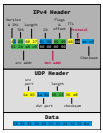


Hav billede af sytem ved siden af
system graf med selve latency
bufferen kan ikke videresende data dirrekte, da den skal gemme segmentet først

Outgoing packet validation:



Evaluation
Validation
Outgoing packet validation:



Protocol ikke sat korrekt, destination ip ikke sat korrekt

Internet Protocol Suite compliancy as per RFC 1122

Evaluation
Validation
Internet Protocol Suite compliancy as per RFC 1122

Ikke testet helt igennem, mem felterne er generelt sat

Table of Contents

Table of Contents

Introduction

Implementation

Evaluation

Discussion

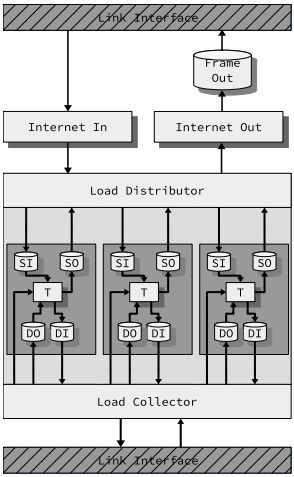
Conclusion

Future Work

Questions

Discussion

Improving the performance:



Estimated performance:

$$1\,\textbf{Byte} * 10\,\textbf{MHz} = 80\,\textbf{Mbps}$$

Discussion
Improving the performance:
Estimated performance:
1 Byte * 10 MHz = 80 Mbps

Table of Contents

Table of Contents

Introduction
Implementation
Evaluation
Discussion
Conclusion
Future Work
Questions

Conclusion

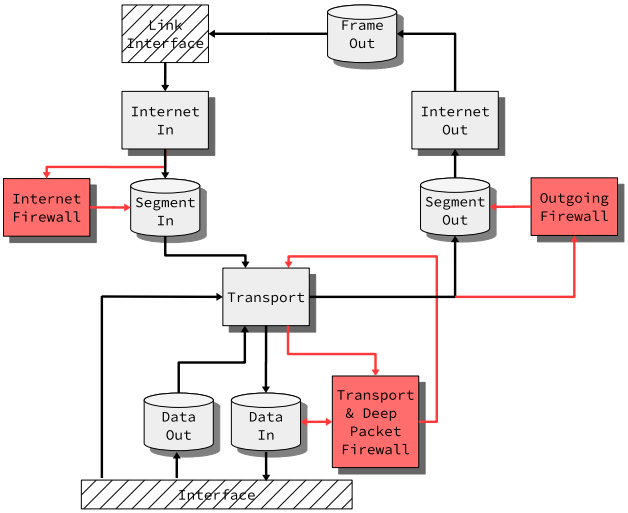
- The design underwent many alternations, but the final layered design has proven to work great
- In 1.83 mio. simulated clock cycles, all of 17283 packets were handled correctly
- Errors in the outgoing packets, but they should be easily fixable
- SME was of great help for the implementation, albeit with a few errors and bugs

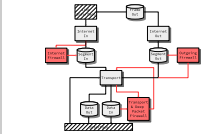
Conclusion
- The design underwent many alternations, but the final layered design has proven to work great
- In 1.83 mio. simulated clock cycles, all of 17283 packets were handled correctly
- Errors in the outgoing packets, but they should be easily fixable
- SME was of great help for the implementation, albeit with a few errors and bugs

Table of Contents

Table of Contents

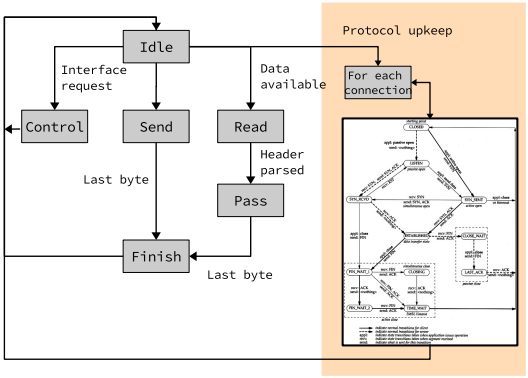
Introduction
Implementation
Evaluation
Discussion
Conclusion
Future Work
Questions





Integration med buffere. Hvad ville det indebære

Implementing TCP



TCP/IP in hardware using SME
└Future Work
└Future Work
2019-09-18

Future Work
TCP
Implementing TCP

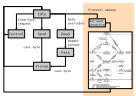


Table of Contents

Table of Contents

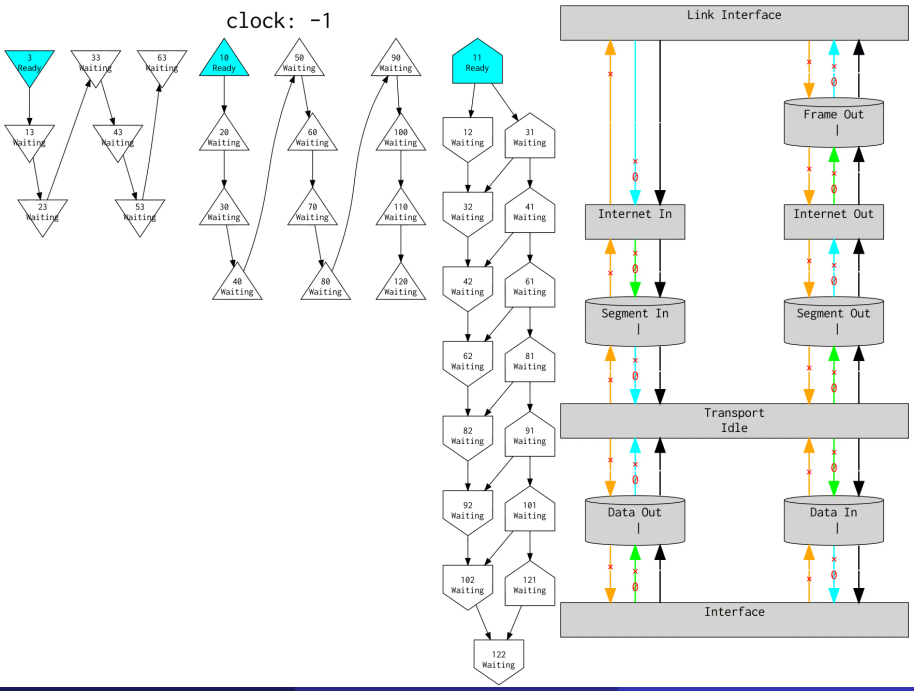
Introduction
Implementation
Evaluation
Discussion
Conclusion
Future Work
Questions

Bibliography

[1] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf. Disaggregated fpgas: Network performance comparison against bare-metal servers, virtual machines and linux containers. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 9–17, Dec 2016. doi: 10.1109/CloudCom.2016.0018.

Bibliography

[1] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf. Disaggregated fpgas: Network performance comparison against bare-metal servers, virtual machines and linux containers. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 9–17, Dec 2016. doi: 10.1109/CloudCom.2016.0018.



2019-09-18

TCP/IP in hardware using SME

└Questions