TCP/IP in hardware using SME

Mark Jan Jacobi & Jan Meznik

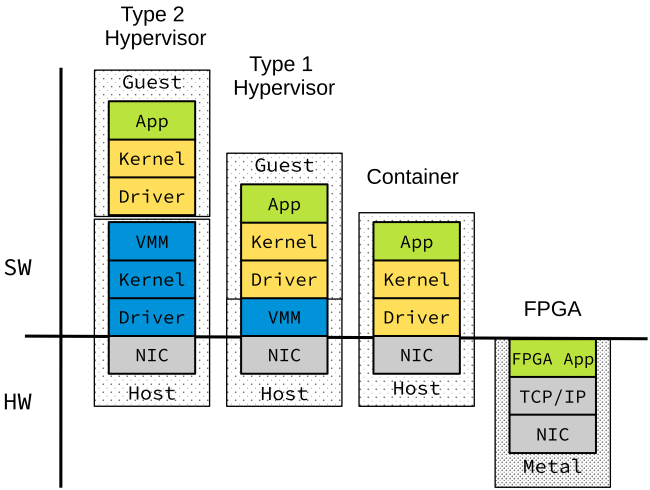
KU

September 11, 2019

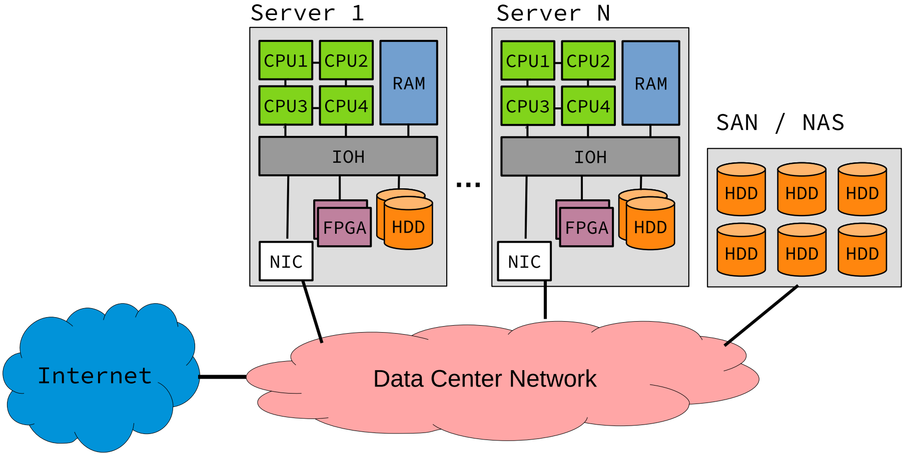
Table of Contents

Background and Motivation

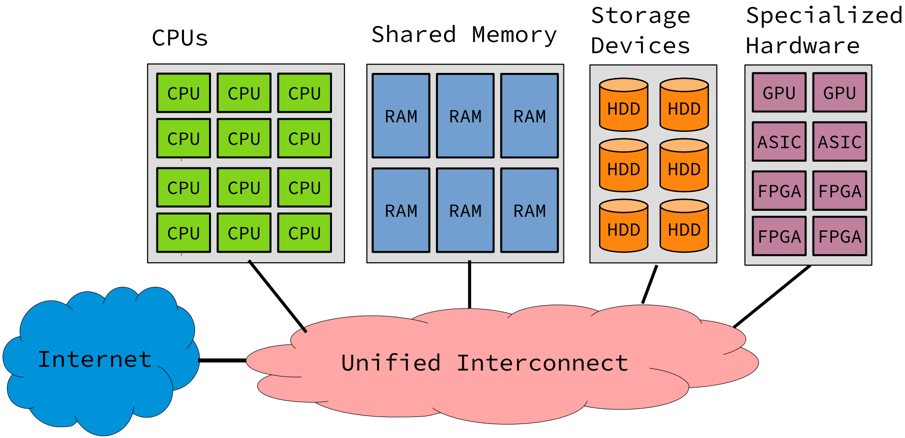
FPGAs are making their way into data centers to boost the computing power and the overall power efficiency.



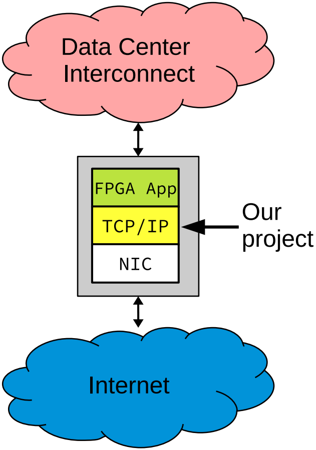
A conventional data center architecture



Proposed, disaggregated data center architecture



FPGA usage



Data Center
Interconnect
FPGA App
TCP/IP
NIC
Our
project
Internet

The Internet What is the internet, and how does it work??!?

TCP/IP 4 layers

Communication example between 2 hosts

Table of Contents

Implementation

- Processes
  - State machines
- Buffers
  - Memory segments
  - Dictionary
- Interface signal control
  - Buffer-Producer
  - Compute-Producer
- Interface control
  - Usage
  - Limitations

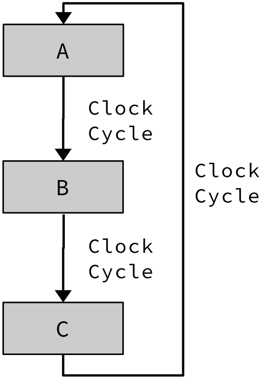
Implementation

Processes

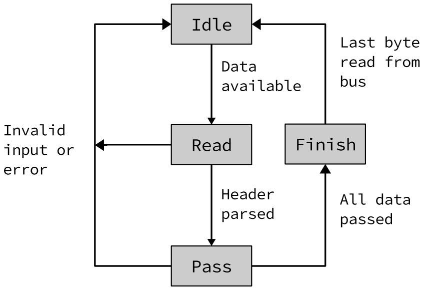
State machines

```
public class SomeProcess : StateProcess
{
  private override async Task OnTickAsync()
  {
    a();
    await ClockAsync();
    b();
    await ClockAsync();
    c();
    await ClockAsync();
  }
}
```

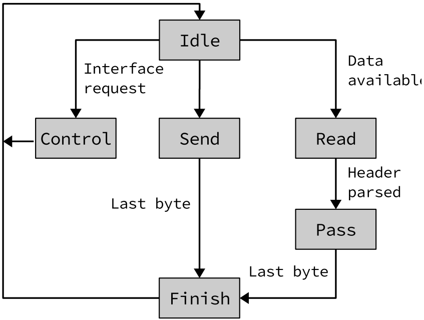
```
public class SomeProcess : SimpleProcess
{
// Initial state
state = A;

protected override void OnTick()
{
  switch(state) {
    case A:
      a();
      state = B;
    case B:
      b();
      state = C;
    case C:
      c();
      state = A;
  }
}
```



Examples



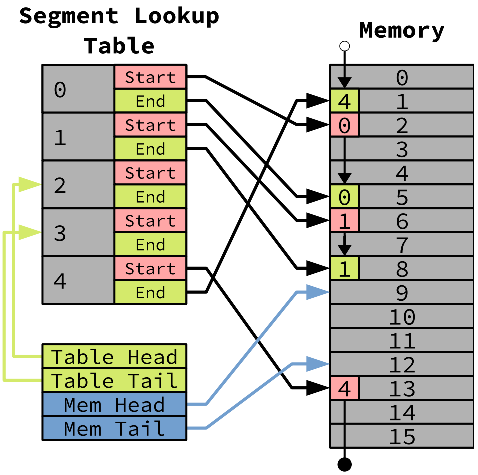
Idle
Data available
Read
Header parsed
Pass
Finish
Last byte read from bus
All data passed
Invalid input or error



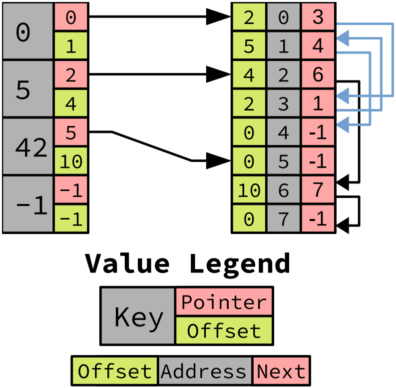
Idle
Interface request
Data availabl
Control
Send
Read
Header parsed
Pass
Last byte
Last byte
Finish

Implementation
Buffers

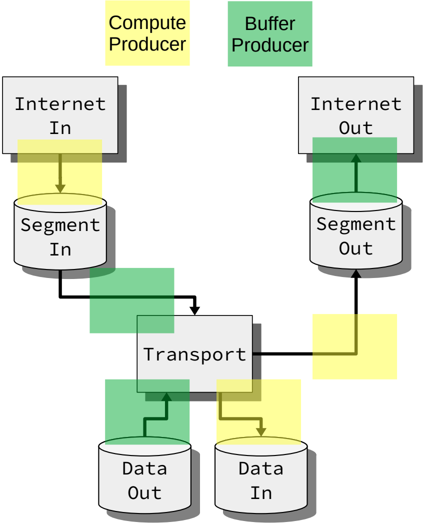
Memory segments



Memory dictionary

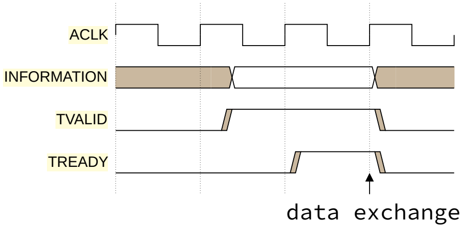


Identifying the scenarios

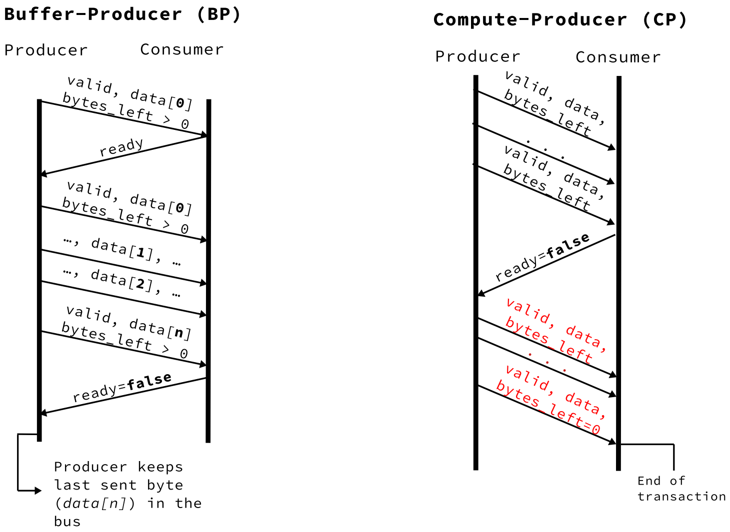


Inspired by AXI4

- Single clock offset when sending data.
- Indicate end of stream with `bytes_left`.



Implementation
Interface signal protocol



The interface structures

```
enum InterfaceFunction : byte
{
    INVALID = 0,
    // BIND = 1,
    LISTEN = 2,
    CONNECT = 3,
    ACCEPT = 4,
    CLOSE = 7,
    // ...
    OPEN = 255,
}

struct InterfaceData
{
    public int socket;
    public uint ip;
    public byte protocol;
    public ushort port;
}
```

```
interface InterfaceBus : IBus
{
    bool valid;
    byte interface_function;
    InterfaceData request;
}

interface InterfaceControlBus : IBus
{
    bool valid;

    byte exit_status;
    byte interface_function;
    InterfaceData request;
    InterfaceData response;
}
```

Limitations

- One request at a time.
- Arbitrary delay between request and response.

Table of Contents

Evaluation

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

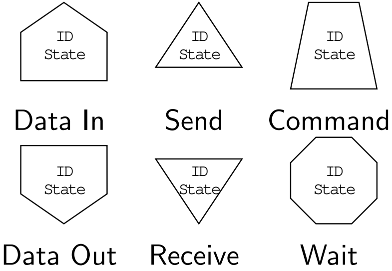
Graph file simulation

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

Evaluation

Setup

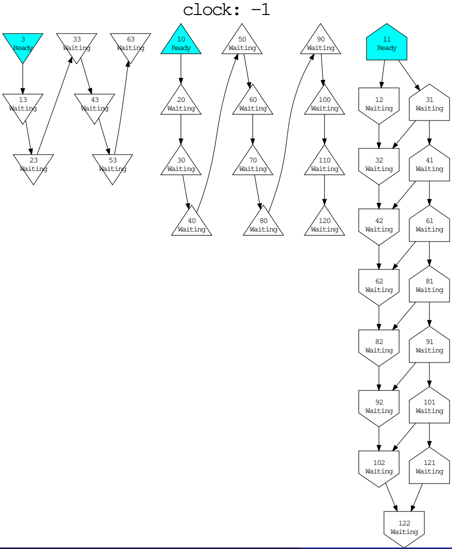
Graph simulation node types

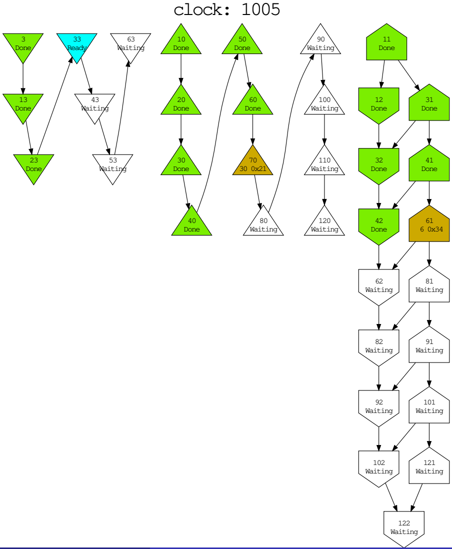


State and Color	Description
Waiting	Vertex is not in use.
Ready	Vertex Is ready for activation.
Active	Vertex is active. Simulator is gathering data.
Inactive	Vertex is inactive. Simulator is not gathering data.
Done	Vertex is done and validated.

Evaluation

Setup





Senario

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

Evaluation
Test

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:

$n_{\mathtt{D}}$: The number of bytes in the data part of the protocol. This excludes both headers from transport and internet.

$n_{\mathtt{I}}$: The internet header size.

$n_{\mathtt{T}}$: The transport header size.

n : The total packet size.

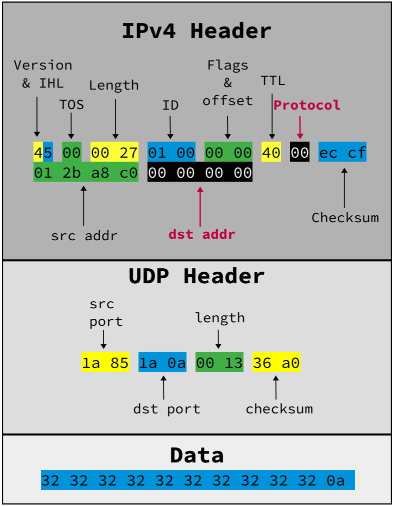
From packet to user

$$6 + n_{\mathtt{I}} + 2n_{\mathtt{T}} + 3n_{\mathtt{D}}$$

From user to packet

$$8 + 2n_{\mathtt{I}} + 3n_{\mathtt{T}} + 4n_{\mathtt{D}}$$

Outgoing packet validation:



Internet Protocol Suite compliancy as per RFC 1122

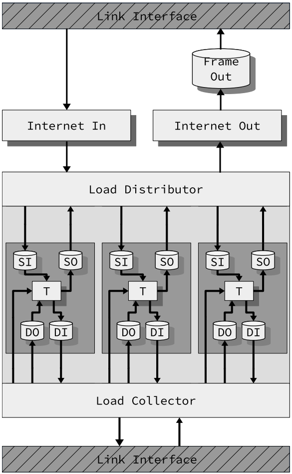
Table of Contents

Discussion

Estimated performance:

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

Improving the performance:



Usability

SOMETHING

Using C#

State modelling
Simulation
Concurrency

Table of Contents

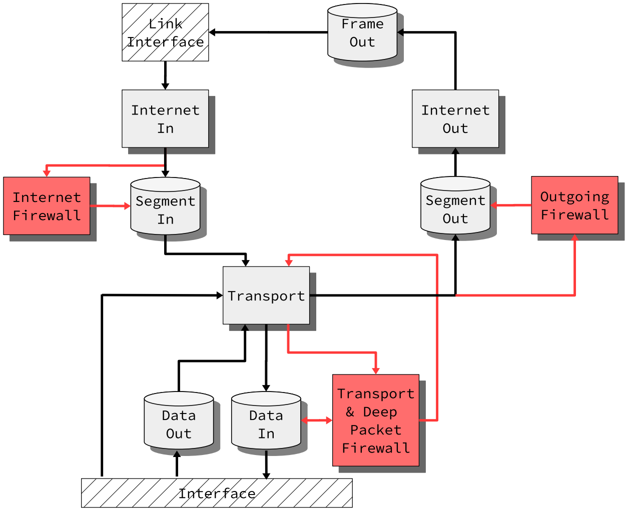
Conclusion

- Lot of trial and error to find the optimal design in the beginning
- In 10 mio. simulated clock cycles, 17283 packets were handled, 1280 of which were correctly received by the Application layer, and then sent out again
- Even with a few flaws, SME is a great framework for hardware modelling

Table of Contents

Future Work

Firewall



Future Work
TCP

How to implement TCP

Table of Contents

Questions

? Some random citation so it does not complain[Andrew S Tanenbaum(2013)]

Bibliography

Todd Austin Andrew S Tanenbaum.
Structured computer organization.
Pearson, Boston, 2013.
ISBN 978-0-273-76924-8.

end

end