Hamburg University of Technology
Institute of Computer Technology

Prof. Dr. F. Mayer-Lindenberg

TUHH
Technische Universität Hamburg-Harburg

FPGA Based Routing Switch Design

ER-4: Hoffman-Singleton Graph Processor Network

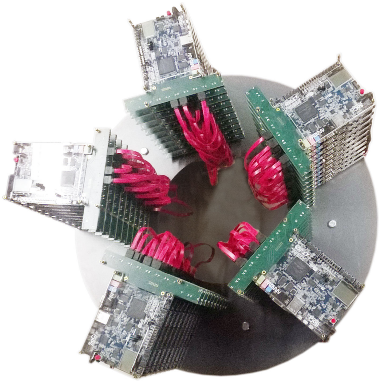
Sergiu Mosanu

January 7, 2015

Experimental Computer ER-4

Introduction

- $5 \cdot 10$ terasIC SoCkit boards
- ARM and FPGA SoC chips
- Multiple interfaces
  - 8 high-speed serial transceivers

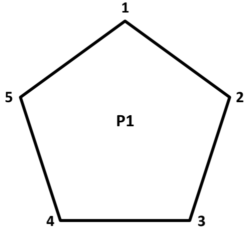


Introduction
and Motivation

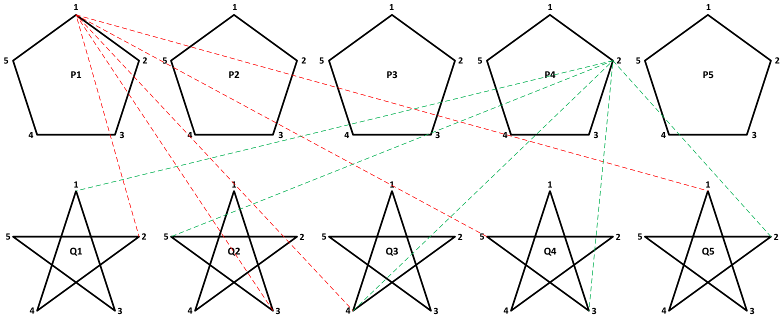
- Project: Experimental Computer ER-4
- Based on: Hoffman-Singleton graph
- Contribution: Routing Switch Module
- Design: Modular and Layered
- Flexibility: Scheduling Policy, Buffer, PRNG parameters, etc.
- Tests: Simulations and Experiments on Xilinx and Altera boards

Graph Theory
Basic Concepts

- Graph G as a set of vertices V (nodes) connected by edges E
- Cardinality (order) $|V|$ and size $|E|$
- Vertex degree $deg(V_i)$ and Regular Graph
- Diameter k and Girth g
- Adjacency and Distance Matrix
- Moore Graph
  - $g = 2k + 1$
  - $|V| = 1 + d\sum_{i=0}^{k-1}(d-1)^i$
- Hoffman-Singleton Graph
  - $d = 7$, $|V| = 50$, $|E| = 175$, $g = 5$, $k = 2$



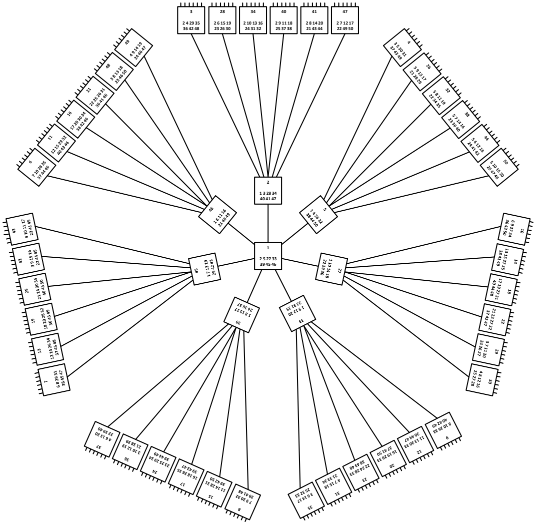
Hoffman-Singleton Graph
Robertson's Construction Method



<u>Rule</u>: vertex i residing in a pentagon P_j connects only with vertex $(i + jk) \bmod 5$ in pentagram Q_k

Hoffman-Singleton Graph

V_1 Edges for hops 1 and 2



Experimental Computer ER-4

Message Structure

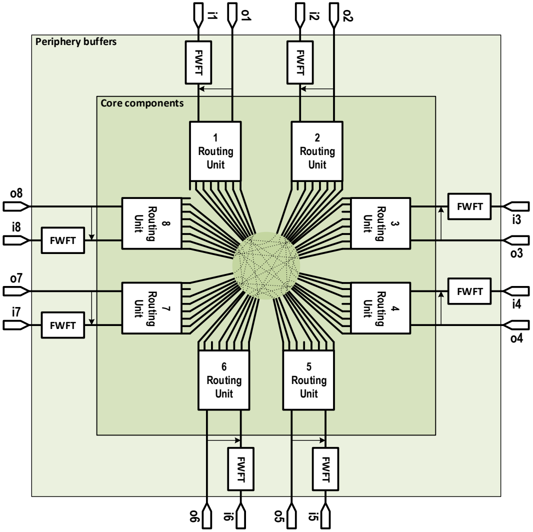
- $18 + 1-$bit words
- Header identifies destination
  - $2 \cdot 4-$bit address for hops 1 and 2
  - $5-$bit bus address for the NoC
- Last word has LSB set to '1'

Index	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
Header	1	**0**	**1**	**1**	1	**1**	**0**	**0**	1	0	1	0	0	0	0	1	1	1	0
Word 1	1	0	0	0	1	1	1	1	0	1	0	0	0	0	0	1	0	0	0
Other words										⋮									
Last Word	0	0	0	0	1	1	1	0	0	1	1	0	1	0	1	1	0	0	**1**

- No network knowledge is stored in any switch!

Switch

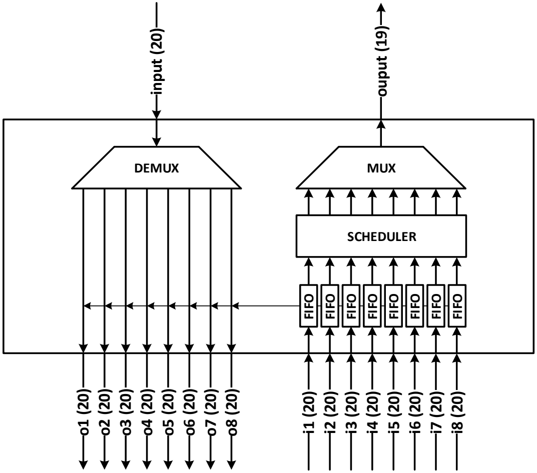
Layers and Modules



Switch
Description

- Simple to test, maintain, develop
- Congestion control (queuing, buffering) and routing
- Periphery FWFT buffers provide additional buffering for internal FIFO buffers
  - Internal buffers' $almost\_full$ flags are checked

Routing Unit
Components and Connections



Routing Unit
Description

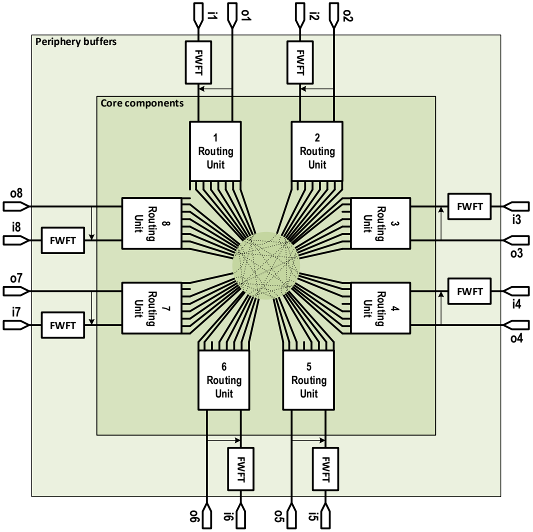
- De-multiplexes input and sends it to desired different routing unit
- Each FIFO buffer provides occupancy information
- Scheduler manages the FIFO buffers reading
- Multiplexer sends the messages to the output
- An almost_full flag is computed for each FIFO buffer as product of the 3 MSB and is appended to each output o_i

Core and Periphery Layer
Implementation details

- Core Layer
  - Passes the almost_full flags and MUX outputs directly to the Periphery Layer
  - Interconnects the Routing Units with duplex channels
- Periphery Layer
  - MUX outputs are further connected to switch's outputs
  - High-speed serial transceivers adapting modules
  - Switch's inputs are summed, detected, and inserted into the FWFT buffers
  - Receiving buffer's availability is judged based on the first word's address and the almost_full fags

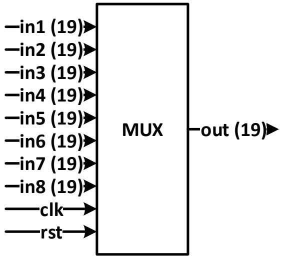
Switch

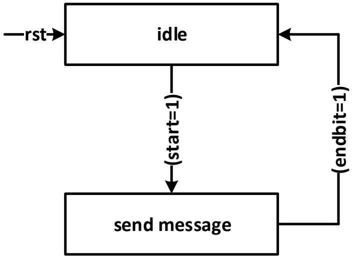
Layers and Modules



Multiplexer Logic

Structure, State Machine, Encoding



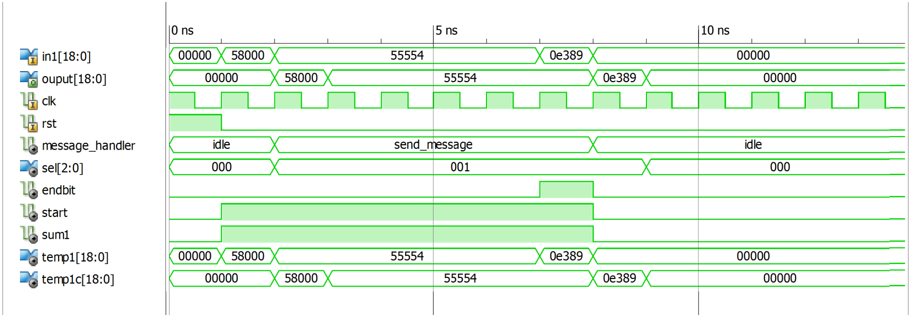


i	sel
1	001
2	010
3	011
4	100
5	101
6	110
7	111
8	000

Multiplexer Logic
Description

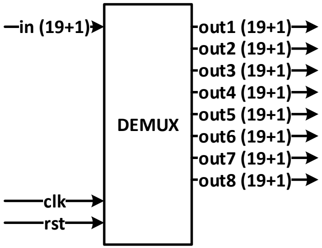
- Lowest level component which finishes the routing and message handling
- Detects start & end of active input and connects input to output
- State machine with state idle and $\mathrm{send\_message}$
- Signal act_i, sel, start
- Signal $\mathrm{endbit} = (\mathrm{act}_1 \cdot \mathrm{in}_1(0)) + \ldots + (\mathrm{act}_8 \cdot \mathrm{in}_8(0))$
- No two simultaneous active inputs
- Overall time delay of one clock cycle

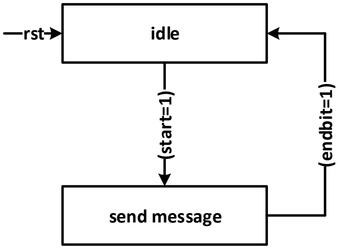
Multiplexer Logic
Simulation



Inverse Multiplexer Logic

Structure, State Machine, Encoding





i	sel
1	001
2	010
3	011
4	100
5	101
6	110
7	111
8	000

Inverse Multiplexer Logic
Description

- Input contain a control bit
- Signal start is connected to the control bit $\mathrm{input}(0)$
- Signal endbit is connected to message's LSB $\mathrm{input}(1)$
- Performs routing and address management
- Signal $\mathrm{sel} = \mathrm{first\_word}(18 \text{ downto } 16)$
- Address Management

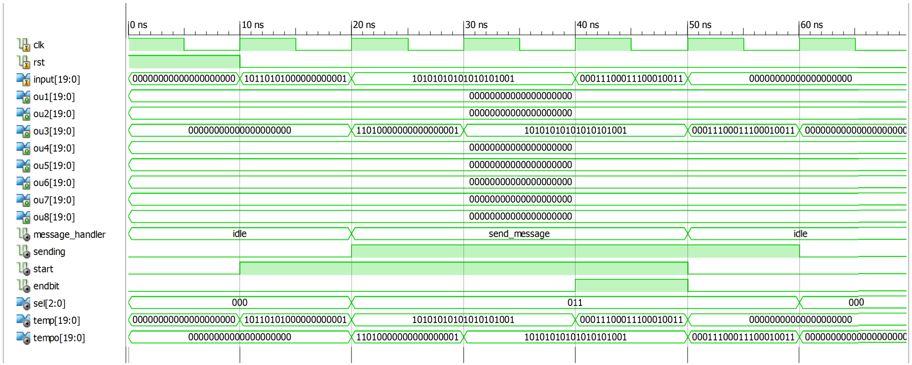
$$\begin{aligned} \mathrm{first\_word}(18 \text{ downto } 16) &= \mathrm{first\_word}(14 \text{ downto } 12) \\ \mathrm{first\_word}(14 \text{ downto } 12) &= (\mathrm{others} = \text{‘0’}) \end{aligned}$$

- Overall time delay of one clock cycle

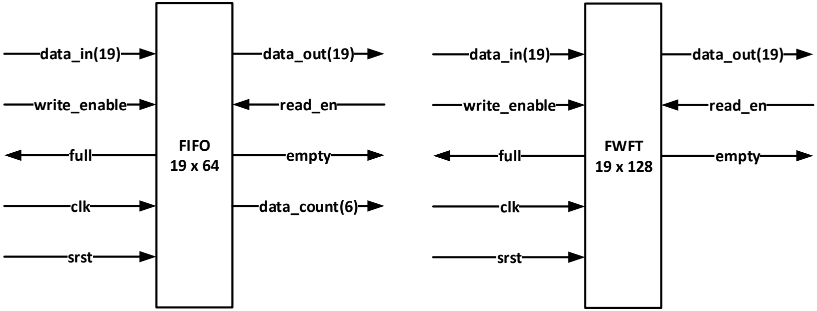
Inverse Multiplexer Logic

Simulation

TUHH Technische Universität Hamburg-Harburg



FIFO and FWFT Buffers
Interface

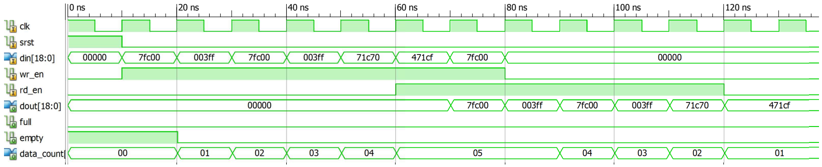


- FIFO: First In First Out Buffer
- FWFT: First Word Fall Through FIFO Buffer

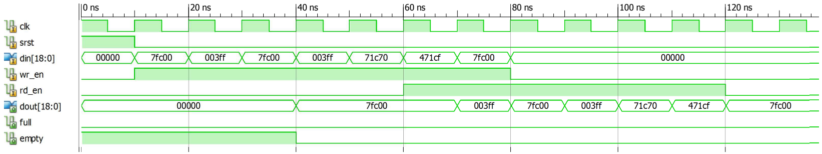
FIFO and FWFT Buffers

Simulation Results

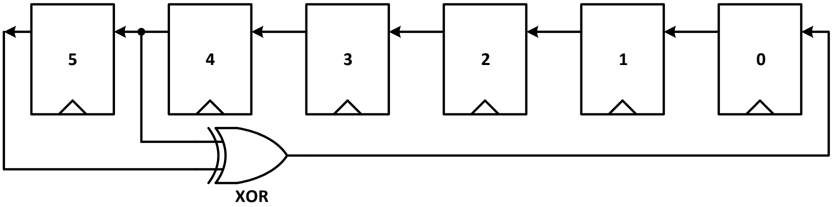
- FIFO simulation



- FWFT simulation

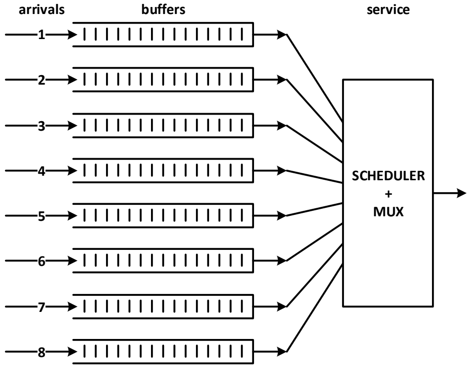


Pseudorandom Number Generator
Linear Feedback Shift Register (LFSR)



- Queuing system of 8 buffers $\Rightarrow 3-$bit random number required
- Used a $6-$bit LFSR module, only bit 2, 4 and 6 were sent out
- Feedback Polynomial is $x^6 + x^5 + 1$, period is $2^6 - 1 = 63$
- Delivers uniformly distributed random numbers

Routing Unit Queuing Logic
Principles



1. M/D/1/512/Random
2. M/D/1/512/SQF_LQF
3. *M/D/1/512/Priority*

M/D/1/512/Random
Description

- Random buffer selection
  - Equal priority for each buffer
- Simple to develop and requires little resources
- Wasteful
  - Failing to guess a non-empty buffer
  - Failing to select a very short or an almost full buffer
- Improvement: select buffer also based on the empty flags

M/D/1/512/SQF_LQF
Description

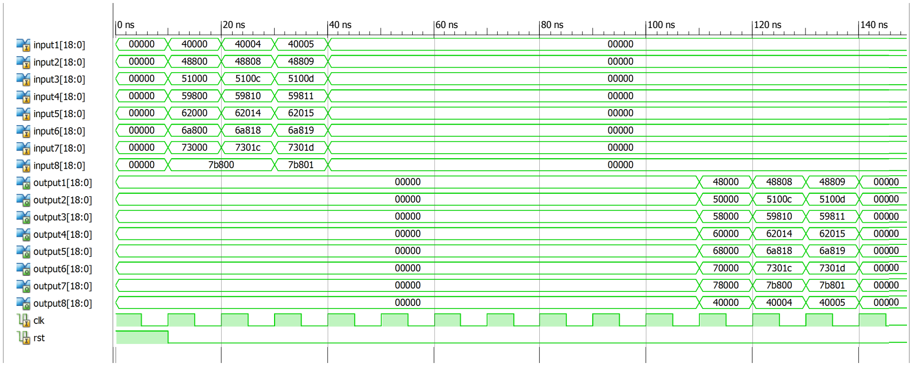
- Based on buffers' occupancy
- When buffers are relatively free
  - the less occupied ones should be processed first
  - very dynamic scheduling
- When one or more buffers are more than half filled
  - the most occupied ones should be processed first
  - overflows are prevented
- Higher performance and reliability
- $\mathrm{SQF\_LQF} = \sum_{i=1}^{8} \mathrm{Occupancy}_i(6)$
- Uses $7 + 6 + \ldots + 1 = 28$ comparators to find largest $\mathrm{Occ}_{i\mathrm{u}}$
- $\mathrm{Occ}_{i\mathrm{u}} = \mathrm{Occ}_i \text{ when } (\mathrm{SQF\_LQF} = \text{`1'}) \,||\, (\mathrm{Occ}_i = \text{`0'}) \text{ else}$
  $= \neg\mathrm{Occ}_i$

M/D/1/512/Priority
Future Possible Improvement

- Sender can set a message's priority, e.g. low, normal, high
- Can be encoded in a message's header
- Buffers can be switched to FWFT or accompanied by priority flags
- More efficient and reliable scheduling policies can be developed based on application requirements

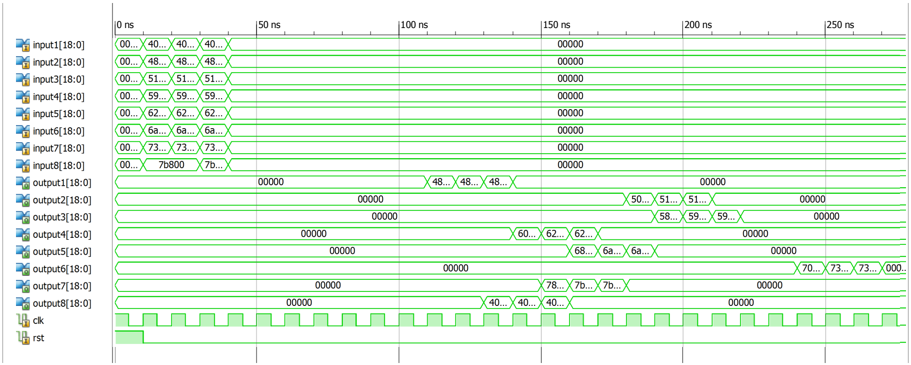
Full Switch using SQF_LQF

Simulation



Full Switch using PRNG

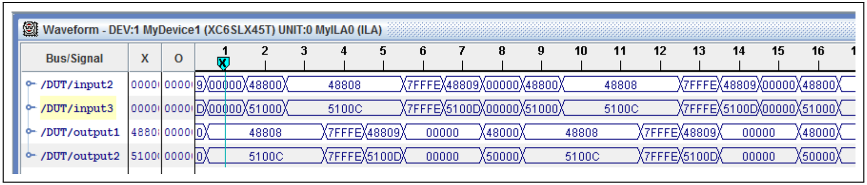
Simulation



Full Switch using SQF_LQF

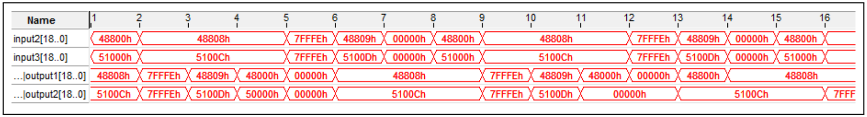
Xilinx and Altera Board Experiment

- Xilinx experiment:



Waveform - DEV:1 MyDevice1 (XC6SLX45T) UNIT:0 MyILA0 (ILA)
Bus/Signal X O 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
/DUT/input2 0000 0000 9 00000 48800 48808 7FFFE 48809 00000 48800 48808 7FFFE 48809 00000 48800
/DUT/input3 0000 0000 D 00000 51000 5100C 7FFFE 5100D 00000 51000 5100C 7FFFE 5100D 00000 51000
/DUT/output1 4880 0000 0 48808 7FFFE 48809 00000 48000 48808 7FFFE 48809 00000 48000
/DUT/output2 5100 0000 0 5100C 7FFFE 5100D 00000 50000 5100C 7FFFE 5100D 00000 50000

- Altera experiment:



Name 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
input2[18..0] 48800h 48808h 7FFFEh 48809h 00000h 48800h 48808h 7FFFEh 48809h 00000h 48800h
input3[18..0] 51000h 5100Ch 7FFFEh 5100Dh 00000h 51000h 5100Ch 7FFFEh 5100Dh 00000h 51000h
...|output1[18..0] 48808h 7FFFEh 48809h 48000h 00000h 48808h 7FFFEh 48809h 48000h 00000h 48800h 48808h
...|output2[18..0] 5100Ch 7FFFEh 5100Dh 50000h 00000h 5100Ch 7FFFEh 5100Dh 00000h 5100Ch 7FFF

Conclusion
and Future Work

- Reliable, Efficient, Cross-platform
- Flexible, Maintainable
- Further improvements
  - Develop an efficient high-speed serial interface adapting module
  - Pipeline troublesome computations to increase operating frequency
  - Develop additional features
  - Further optimize design

Conclusion
and Future Work

- Reliable, Efficient, Cross-platform
- Flexible, Maintainable
- Further improvements
  - Develop an efficient high-speed serial interface adapting module
  - Pipeline troublesome computations to increase operating frequency
  - Develop additional features
  - Further optimize design

Thank You!