Accelerator-in-Switch: a novel cooperation framework for FPGAs and GPUs
— Building a Virtual Heterogeneous Computing System —

2018.12
Hideharu Amano
Keio University

This work is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO)
FPGA computing in cloud

- High Performance FPGAs are available:
  - High computational power: Intel Stratix-10
  - With a large memory: Xilinx UltraScale+ with UltraRAM
  - But they are expensive for most users to keep themselves.

- Programming environment is improved:
  - Open-CL is widespread for computational usage.
  - Vivado-HLS is popularly used for general usage.

→ FPGAs in cloud:
  More flexible and power efficient than using GPU.

- FPGA in the Cloud: Booting Virtualized Hardware Accelerators with OpenStack [FCCM2014]
- Microsoft Catapult [ISCA2014 ][HEART2017]
- FPGA Supervessel Cloud by IBM[ICFPT2016]
- Amazon EC2 F1 Instance [https://aws.amazon.com/ec2/instance-types/f1/]
Conventional FPGA-in-Cloud

High-Performance FPGAs are attached into each Host.

- The total cost becomes large.
  - High performance FPGAs are still expensive.
- The size of FPGA is limited.
  - Multiple FPGAs cannot be used together.
Our Proposal: Virtual Large FPGA

A lot of cost-efficient middle-scale FPGAs are tightly connected. They can be treated as if they were a single FPGA in HLS description level.

Higher performance per cost than conventional FPGA in cloud. Practically infinite resource is used. Separated into a number of virtual FPGAs and shared by the multiple users.

Flow-in-Cloud (FiC) is the first prototype.
Microsoft’s Catapult V1/V2

[Putnum : ISCA-2014][Caulfield : Micro-2016]

Rank computation for Web search on Bing.
Task Level Macro-Pipelining (MISD)
FE: Feature Extraction
FFE: Free Form Expression: Synthesis of feature values
MLS: Machine Learning Scoring

2-Dimensional Mesh is formed (8x6) for 1 cluster.

10Gbps network is upgraded to 40Gbps network in V2
Recent FPGA supercomputers (For example Riken’s)
Today’s talk

• Building a virtual large FPGA
  • Concept 1: Use middle-range FPGAs and common serial links
  • Concept 2: Virtualize at the level of HLS description
  • Concept 3: Couple accelerators and a switch tightly in an FPGA
    → Accelerator-in-Switch

• Our prototype: FiC (Flow-in-Cloud)

• Next step: Building a virtual heterogeneous computing system
Here, we will omit “XC”.
KU085 means XCKU085.
1. Multiple middle scale FPGAs vs. a single powerful FPGA

- Most of price/resource of KU085 is the lowest.
- Two KU085s can provide 1.5 LCs of KU115 with almost the same price.
- Five KU085s can provide almost the same LCs of VU440 with about 1/3 price.

### Comparison Table

<table>
<thead>
<tr>
<th></th>
<th>Kintex Ultrascale KU085</th>
<th>Kintex Ultrascale KU115</th>
<th>Virtex Ultrascale VU440</th>
<th>Virtex Ultrascale+ VU9P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Cell (K)</td>
<td>1088 (3.4)</td>
<td>1451(4.3)</td>
<td>5541(10.6)</td>
<td>2586(6.69)</td>
</tr>
<tr>
<td>DSP</td>
<td>4100(0.9)</td>
<td>5520(1.14)</td>
<td>2880(20.4)</td>
<td>6840(2.53)</td>
</tr>
<tr>
<td>BRAM(Mb)</td>
<td>56.9(65.4)</td>
<td>75.9(83.0)</td>
<td>88.6(664.6)</td>
<td>345.9(50.05)</td>
</tr>
<tr>
<td>Price ($)</td>
<td>3720</td>
<td>6297</td>
<td>58890</td>
<td>17314</td>
</tr>
</tbody>
</table>

Price is from digikey
( ) is price for each unit.
Price/KLC is increased with its size, because high-end FPGAs have special facilities.
Kintex Ultrascale has surprisingly large number of DSPs
BRAM vs. Price is almost linear, but Kintex Ultrascale is cost effective.

Virtex Ultrascale+ has a large memory by UltraRAM but expensive.
Cost including serial links

• KU085 has the best price per resource!
  • Logic cells: $3.4 for 1K LC.
  • DSP: $0.9 for a DSP.
  • BRAM: $65.4 for 1Mb.
• Let’s use a number of common serial links GTH (12.5Gbps).
  • Of course, faster serial links (32Gbps GTY, 58Gbps GTM) are available, but cost becomes high.
• Firefly cable ($59 for 4 links) is available.
  • No drivers/receivers are needed.
  • Aurora IPs from Xilinx can be used.

• Conclusion: “Using multiple middle scale FPGAs” is a cost efficient solution.
  • Open issue
    • Cost of switches
    • Operational Speed
2. Virtualization at the IP based HLS design

A single FPGA

HLS is originally described with a set of IPs: Division is easy except the problem of handshake.
The handshake problem

- Valid signal can be omitted by checking the data arrival.
- Without a ready signal, the possibility of input FIFO overflow remains.
Overriding the handshake problem

- **Virtual ready wire**: providing a virtual wire between the receiver and the sender.
  - Direct approach but the overhead of synchronization may increase.
- **Providing a required memory inside HLS module**.
  - Convenient for streaming processing but the HLS programmer must take care of it.
- **A pre-processor can insert delay or synchronization code according to the evaluation results from Vivado HLS.**

→ All methods require fixed latency/throughput communication. Our approach: circuit switching with Static Time Division Multiplexing.
Vivado HLS evaluates the number of clock cycles in a loop. → Such information helps override the handshake problem.
An example: Streaming processing

HLS 1

200 data / 5000 clock cycles

Delay

HLS 2

200 data / 5000 clock cycles

FIFO OVF

HLS 3

200 data / 10000 clock cycles

HLS 4

100 data / 10000 clock cycles

HLS 5

Delay

FIFO

Delay

HLS 1

200 data / 10000 clock cycles

HLS 2

200 data / 10000 clock cycles

HLS 3

200 data / 10000 clock cycles

HLS 4

100 data / 10000 clock cycles

HLS 5
3. Integrating switch and accelerator tightly in an FPGA: Accelerators in Switch

• What is the benefit of FPGAs compared to GPUs?
  • Switching capability is much superior to that of GPUs.
  • Of course, recent GPUs provide NVLinks or other powerful interface.
    • However, they are only for expensive GPUs, and the function is limited.
  • Various type of switches can be implemented on FPGAs.
  • FPGAs are widely used for high speed switches and network interface.

• Tightly coupled switch and accelerator in an FPGA.
• Separation with Partial Reconfiguration
  → Accelerator in Switch [FPL2017]
An example of AiS (PEACH3)

- Implemented as a module on the Avalon MM bus.
- Shared memory is used for exchange of data.
- Reduction / Locally Essential Tree generation were implemented in the AiS part.
The execution time for LET generation
Today’s talk

- Building a virtual large FPGA
  - Concept 1: Use middle-range FPGAs and common serial links
  - Concept 2: Virtualize at the level of HLS description
  - Concept 3: Couple accelerators and a switch tightly in an FPGA
    → Accelerator-in-Switch
- Our prototype: FiC (Flow-in-Cloud)
- Next step: Building a virtual heterogeneous computing system
Flow-in-Cloud (FiC) overview
Here, we call each link “channel”, and a bundle of 4 channels “lane”.

A board has 8 lanes each of which has 4 channels.
Block Diagram of FiC

- **DRAM**
- **PR domain**
  - 100MHz
- **Static domain**
  - Xilinx Aurora
  - 170bit
  - 9.9 Gbps
  - 8.5 Gbps x32 (4 chan. x 8 lane)
- **Ethernet**
- **Raspi3**
- **HLS module**
- **STDM switch**
  - 170bit
  - 9.9 Gbps
  - 8.5 Gbps x32 (4 chan. x 8 lane)
STDM (Static Time Division Multiplexing)

An input register is selected according to the pre-loaded table, and transferred to the output register.

Input data arrive at each port cyclically registered.

Output data are cyclically sent to the output port.

An example of 4x4 with four slots
STDM (Static Time Division Multiplexing)

- A circuit is established between source and destination.
- Latency and bandwidth are kept.

Latency = 55 + 2 x (# of slots) clock cycles
Multicast using the STDM

Multicast is done efficiently.

Multiple outputs can receive the same data in a specific slot.
The resource usage

4 switches are provided for each channel.

GT: High speed link

Enough design is remained for HLS design.
How boards should be connected?

• Any type of interconnection is OK.
• However, there are two limitations:
  • 4 channels are bundled into a lane.
  • For HLS modules, the size should be less than four 9x9 switches.
• 4 channels in a lane are used independently.
  • An HLS module has four independent ports, or four HLS modules with a port are implemented at maximum.

Network with 8-degree
→ Natural solution: 4 dimensional torus
  The diameter is large.
→ Alternative: Full mesh Connected Cycles (FCC)
  Dragonfly-like network but more economical.
4-DTorus: the case of \(3 \times 3 \times 3 \times 3 = 81\) boards

Suitable if local traffic is dominant.
Diameter is relatively large: 8
4x24 Full mesh Connected Cycles (FCC)

24 Cycles are connected in full-mesh

96 boards are connected
Estimation of the network performance

<table>
<thead>
<tr>
<th></th>
<th>3x3x3x3 Torus</th>
<th>4x24 FCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of boards</td>
<td>81</td>
<td>94</td>
</tr>
<tr>
<td>Slots (Bit complement/tornado/reversal)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Slots (All to all)</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>Diameter</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Max Latency (All to all nsec)</td>
<td>5360</td>
<td>3550</td>
</tr>
</tbody>
</table>
Topology Optimization for Traffic Pattern*

Recursive Partition

Circuit Switching

Generated topo. with # of slots

Result: comparison with mesh

# of switches reduced by up to 56.3%
Avg. hop count Reduced by up to 83.7%

Partial Reconfiguration for separating HLS from switches.

Switch/Link synthesis

HLS module 1 synthesis

Setting the PR region

opt_design, place_design, route_design

HLS1 bitmap generation

Make PR region into blackbox
Lock wires in the static region

Design of HLS modules can be done only with this part.

HLS module 2 synthesis

HLS module 3 synthesis

Read into the blackbox

Basic Design

opt_design, place_design, route_design

HLS2 bitmap generation

opt_design, place_design, route_design

HLS3 bitmap generation
Now, 3 lanes (12 channels) are used.
The current FiC system

- Users
- FPGA Configuration and Table information (*.json)
- Deliver of Configuration data with RESTful API through HTTP
- Results
- Control Server
- Internet / Intranet
- GUI control from remote terminals
- FiC Board Control Network
- FiC Network
Flow-in-Cloud Board (on fic08)

GUI from remote terminals

FIC hostname: fic08

**FPGA**

**Status**

- **Status:** Status at Sat, 17 Nov 2018 10:14:10 GMT

**Control**

- FPGA RESET
- HLS MODULE RESET
- HLS MODULE START

**Configure**

You can upload FPGA *.bit file from here.

**Configuration mode:**

- Selectmap x16
- Selectmap x16 (PR)
- Selectmap x8
- Selectmap x8 (PR)

**Browse:** C: \fakepath \Xtrst.bin

**Upload** Burn FPGA... wait 83s...
Implementation Example (LeNet Fully Connected layer)

Input buffer (x2)

- input[0]
- Input[1]
- ...
- input[M-1]

Weight buffer

- weight[0][0]
- weight[0][1]
- ...
- weight[N-1][M-1]

Register

Output buffer (x2)

- output[0]
- output[1]
- ...
- output[N-1]

Bias buffer

- bias[0]
-...
- bias[N-1]

ReLu

Size (N, M) multiply-add

M × N × M

Register Input buffer (x2)

PE[0]

PE[1]

PE[N-1]
Fully connection layer of the Lenet

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Power (W)</th>
<th>image/sec</th>
<th>GOPS</th>
<th>GOPS/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 board</td>
<td>100</td>
<td>17.89</td>
<td>23551</td>
<td>120.58</td>
</tr>
<tr>
<td>4 boards</td>
<td>100</td>
<td>71.56</td>
<td>94226</td>
<td>482.34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Power (W)</th>
<th>GOPS</th>
<th>GOPS/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>FiC 4boards</td>
<td>100</td>
<td>71.56</td>
<td>482.34</td>
</tr>
<tr>
<td>Stratex-V [1]</td>
<td>120</td>
<td>25.8</td>
<td>136.5</td>
</tr>
<tr>
<td>KCU060 [2]</td>
<td>200</td>
<td>25.0</td>
<td>172.0</td>
</tr>
</tbody>
</table>

Higher performance is achieved with the similar power efficiency compared with a single FPGA system.

Today’s talk

• Building a virtual large FPGA
  • Concept 1: Use middle-range FPGAs and common serial links
  • Concept 2: Virtualize at the level of HLS description
  • Concept 3: Couple accelerators and a switch tightly in an FPGA
    → Accelerator-in-Switch

• Our prototype: FiC (Flow-in-Cloud)

• Next step: Building a virtual heterogeneous computing system
Next Step: Accelerator bare-metal cloud
FLOW-OS is now under development by the national institute of industrial science and technology (AIST)
Conclusion

- A virtual large FPGA:
  - Scalable performance with
  - similar power/performance and
  - smaller cost/performance compared to conventional FPGA-in-clouds.

- Future direction
  → Accelerator bare metal cloud
  Integration of FPGAs and GPUs.