Next Generation
Massively Parallel VLSI Architectures and Design Methods

Ahmed Hemani - Presenter
KTH – Dept of Electronics and Embedded Systems, School of ICT, KTH.

Acknowledgement:
Nasim Farahini, Syed Asad Jafri, Li Shuo, Hassan Sohofi, Muhammad Ali Shami, Adeel Tajammul, Omer Malik,
Problem Analysis
Problem 1: Large Engineering Cost

Engineering: 40 MUSDs
Manufacturing: 10 MUSDs

1. More SOCs will be financed by VCs
2. More power efficient designs
3. New Product categories will emerge
Problem 2: The Energy Efficiency of General Purpose Processors

Over 90% energy of general-purpose von Neumann processor is “overhead”

Source: Eric S. Chung, Peter A. Milder, James C. Hoe, Ken Mai, Computer Architecture Lab at CMU
The Generality Problem Explained

Generality comes at a huge cost of Silicon, Computational and Engineering efficiencies

Custom solutions are Orders of magnitude more efficient
## Comparison of Computational & Silicon Efficiencies

<table>
<thead>
<tr>
<th>Device</th>
<th>GFLOP/s actual</th>
<th>(GFLOP/s)/mm² norm. to 40nm</th>
<th>GFLOP/J norm. to 40nm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MMM</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU-Core i7</td>
<td>96</td>
<td>0.50</td>
<td>1.14</td>
</tr>
<tr>
<td>GPU-GTX480</td>
<td>541</td>
<td>1.28</td>
<td>3.52</td>
</tr>
<tr>
<td>GPU-GTX285</td>
<td>425</td>
<td>2.40</td>
<td>6.78</td>
</tr>
<tr>
<td>GPU-R5870</td>
<td>1491</td>
<td>5.95</td>
<td>9.87</td>
</tr>
<tr>
<td>FPGA-LX760</td>
<td>204</td>
<td>0.53</td>
<td>3.62</td>
</tr>
<tr>
<td>Same RTL in 65nm</td>
<td>694</td>
<td>19.28</td>
<td>50.73</td>
</tr>
<tr>
<td><strong>FFT-1024</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU-Core i7</td>
<td>67</td>
<td>0.35</td>
<td>0.71</td>
</tr>
<tr>
<td>GPU-GTX285</td>
<td>250</td>
<td>1.41</td>
<td>4.2</td>
</tr>
<tr>
<td>GPU-GTX480</td>
<td>453</td>
<td>1.08</td>
<td>4.3</td>
</tr>
<tr>
<td>GPU-R5870</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FPGA-LX760</td>
<td>380</td>
<td>0.99</td>
<td>6.5</td>
</tr>
<tr>
<td>Same RTL in 65nm</td>
<td>952</td>
<td>239</td>
<td>90</td>
</tr>
</tbody>
</table>

Source: Eric S. Chung, Peter A. Milder, James C. Hoe, Ken Mai, Computer Architecture Lab at CMU
Problem 3: Area is Free but Power is Not

2009 Intl. Technology Roadmap for Semiconductors

Normalized to 40nm (log)

- Area density
- Supply voltage
- Power (device)

Moore’s Law

16X area density

Only 4X Lower Power

Source: Eric S. Chung, Peter A. Milder, James C. Hoe, Ken Mai, Computer Architecture Lab at CMU
Those who do not move, do not notice their chains.
Proposed Solution: A Parallel Distributed Customisable Coarse Grain Reconfigurable Fabric
DRRA – Computational Fabric

Dynamically Reconfigurable Resource Array

This is only a fragment

22 nm, 100 mm²
10 000 DRRA Cells

DPU & Register File Outputs
3 Columns to the Left and and to the Right
And this 3 column window slides
Distributed Memory Fabric – DiMARCH

- DiMARCH
- Memory banks
- Instruction NOC Packet switched
- Data NOC Circuit Switched
- Streaming Register Files
- ALU
- Sequencer
- Interconnect fabric
- DRRA
- Computation fabric
Private Execution Partitions

Time Division to Space Division Multiplexing of Computer Resources

Memory Banks can be clustered to serve as one large bank
Programmed to stream data
Can be connected to clusters in computational fabric
The Hardware Centric Platform

Parallelism & Customisation is hardware like and not constrained by design time decisions

**Parallelism**
- Task Level
- Thread Level
- Micro-thread level
- Loop Unrolling
- Data Level

**Distributed Processing**
- Arithmetic
- Address generation
- Control – short wires

**Arbitrarily complex, wide and deep datapath**
- Compile time and run time Customisation instead of design time
So – Does it Work?
Industrial Case Study

40 GOPS/Watt Multi-mode CGRA Accelerator for a Multi-Standard Base-station
Design Methodology
Standard Cells

Abstraction
Boolean level abstraction
Hides circuit and physical design details
Enabled logic synthesis

Physical Design Discipline
Standard pitch and Row based layout
Enabled physical design automation

Improves efficiency of
1. Synthesis from RTL to GDSII
2. Verification at RTL
Problems with the Standard Cells

- The Big Design Space Problem
- The Big Abstraction Gap Problem

FSMD = FSM + Datapath

Abstraction Level

# of Solutions

- Physical Synthesis
- RTL/Logic Synthesis
- High Level Synthesis

(Standard-Cell Design)

μ-architecture

Function Implementations (FSMDs)

System (Hierarchy of Algorithms)
Faking System Level Synthesis
Using typical commercial HLS

Manual Refinement - System Design
1. Budget constraints and Estimate cost metrics
2. Create global interconnect, buffer and control
3. Partition system-model into synthesizable design units

Functional verification:
Partitioned and synthesizable design vs. Golden System-model

Golden System-model
Modem / Codec:
Hierarchy of algorithms

System-level constraints:
Sampling Rate, Total Latency

Constraints verification:
Cost metrics of physical design vs. Estimated cost metrics

Synthesis:
High-level/Logic/Physical

Costly Iterations
State of the Art SOC Design Flow
What is wrong with it?

System: Multiple applications

Compose, Map & Dimension:
1. HW/SW Partitioning
2. Interface Design
3. Memory & Interconnect Hierarchies
4. I/O Design

Architecture Definition in terms of pre-designed IPs

Stitch Architecture: Buy and Assemble

Logic: Algorithm + RTL + Boolean

Automatic High-level
RTL / Logic 
Physical Synthesis

Chip

Functional/Logic Verification

Constraints Verification:
Timing/Energy/Power/Area

Manual
Proposed Solution
An Analogy
Standard Cells as building blocks are not scalable for 10-100 million gate designs.

~10-100 Million gates
The First Proposition – Raise Abstraction to \( \mu \text{Arch level} \)

4-5 orders larger than Standard Cell

SiLago Block

(Register Files, DPUs, Switch boxes, Processors, SRAM banks etc.)

Characterised boolean operations

Characterised micro-architectural operations
Second Proposition:  
New Grid based Physical Discipline

1. A Grid
2. Regions
3. SiLago Blocks
A grid based structured layout scheme

Traditional SOC

1 2 3 4
9 6 5
8 7

SiLago Fabric based SOC

DiMArch: Streaming Data Storage

DRRA: Streaming DSP

System Control

Program Storage

Flexilators

Memory Control

Sensors

Power Mngmt

PLL + CGU

RF/Analog

RF/Analog
SiLago Interfaces

Global
1. Connects regions, sub-regions, SiLago blocks
2. Control and Configure – packet switched NOC
3. Data – circuit switched NOC

Regional
1. Communication among SiLago blocks of the same type
2. Clustering for capacity and degree of parallelism
SiLago Interconnects are also hardened

The SiLago interconnect are not just logical interconnect, i.e., soft.

They are physical in a templateized or parametric manner
SiLago fabrics are composed by abutment

1. SiLago blocks absorbs
   a) Local Clock Tree & Power Ring
   b) Absorbs regional and global interconnect
   c) Pins on the periphery at right positions

2. Fabric Composition by abutment
SiLago Platform → Methodology

Measurement level accuracy of μ-Arch operations: compute, storage and interconnect

Regularity → scalable engineering effort, μ-Arch operations are space invariant

High-level synthesis compose solutions in terms of μ-Arch operations → FSMDs

System-level synthesis compose solution in terms FSMDs + μ-Arch operations
SiLago Sub-System-level Synthesis

1. Select Optimal Solution from $M^L$ solutions
2. Global Interconnect, buffers and control
3. Floorplanning

Number and types of SiLago blocks + Mapping

Compose GDSII Macro

GDSII Macro

Reports
Why does the proposed Solution Work?
How does the SiLago Platform enable Efficient Sub-System Level Synthesis?

- **Abstraction Level**
- **# of Solutions**

The Reduced Design Space

**Reduced Abstraction Gap**

- Function Implementations (FSMDs)
- System (Hierarchy of Algorithms)

Raise the abstraction level of the Physical Design Target
How does the SiLago Platform enable Efficient Sub-System Level Synthesis?

The Reduced Design Space

Reduced Abstraction Gap

Function Implementations (FSMDs)

System (Hierarchy of Algorithms)

Abstraction Level

<table>
<thead>
<tr>
<th># of Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexilator</td>
</tr>
<tr>
<td>Flexilator</td>
</tr>
<tr>
<td>PIL + CGU</td>
</tr>
<tr>
<td>PMC</td>
</tr>
<tr>
<td>Memory Controller</td>
</tr>
<tr>
<td>Streaming Data Storage</td>
</tr>
<tr>
<td>System Control</td>
</tr>
<tr>
<td>Program Storage</td>
</tr>
<tr>
<td>Data Storage</td>
</tr>
</tbody>
</table>
How does it fix the fake Sub-System-level Synthesis?

Golden Sub-System-model:
Modem / Codec
Hierarchy of algorithms

System-level constraints:
Sampling Rate, Total Latency

Manual Sub-System Design
1. Budget constraints and Estimate cost metrics
2. Create global interconnect, buffer and control
3. Partition system-model into synthesizable design units

Functional verification:
Partitioned and synthesizable design vs. Golden System-model

Sub-System-level Synthesis

Constraints verification:
Cost metrics of physical design vs. Estimated cost metrics
How SiLego will Solve SOC-level Synthesis?

**System: Multiple applications**

**System Design**
1. Pins, Package, Board Level issues.
2. HW/SW. Accelerators. Interface
3. Cache, Memory & Interconnect
4. Power Management

**Architecture Definition**
pre-designed IPs

**Stitch Architecture**
RTL + TLM

**Logic: Algorithm + RTL + Boolean**
High-level, Logic & Physical Synthesis

**Synthesis**
Chip / GDSII

**Constraints Verification:** Area, Latency, and Energy

**Software Design**

**Automated**

**Manual**

**Automated**

**Functional Verification**
Experimental Proof that the proposed Solution Works
**SiLago FSMD Library**

**Development Efficiency**

---

**Synthesis Runtime (Seconds)**

100-1000X Better

- Physical Synthesis
- Logic Synthesis
- High-level Synthesis

---

**Energy Estimation Error**

- 1.69%
And what do we pay for it?

**Area Overhead**

<table>
<thead>
<tr>
<th></th>
<th>SiLago</th>
<th>Standard Cell based Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy</td>
<td>1.2</td>
<td>1.0</td>
</tr>
<tr>
<td>Overhead</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Energy Overhead**

<table>
<thead>
<tr>
<th></th>
<th>SiLago</th>
<th>Standard Cell based Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy</td>
<td>1.2</td>
<td>1.0</td>
</tr>
<tr>
<td>Overhead</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This overhead is less than what we paid when we moved to Standard cells from full custom design.

Full custom SiLago Blocks
Better design quality compared to Standard Cells
Foundry model will be violated.
SiLago provides significant improvement in Predictability
Design Space Exploration: SiLago vs Standard Cells

<table>
<thead>
<tr>
<th>Abstraction Level</th>
<th># of Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Level Synthesis</td>
<td>High Level Synthesis</td>
</tr>
<tr>
<td>Gate Level (Standard-Cell Design)</td>
<td>Gate Level (Standard-Cell Design)</td>
</tr>
<tr>
<td>System Level (Hierarchy of Algorithms)</td>
<td>System Level (Hierarchy of Algorithms)</td>
</tr>
</tbody>
</table>
Design Space Exploration in SiLago System-level Synthesis

- **Number of Solutions evaluated by SiLago SLS**
  - JPEG Encoder: 90
  - WLAN Tx: 80
  - LTE Uplink: 30

- **Time required for DSE**
  - 0 to 225 seconds

The diagram illustrates the number of solutions evaluated by SiLago SLS for different tasks and the time required for DSE.
Conclusion

• CGRA enables hardware like efficiency and retains flexibility of software
• Raising the physical design target from boolean level standard cell to micro-architecture level enables efficient high-level and system level synthesis
• System level synthesis eliminates the verification cost by virtue of correct by construction
Thanks for your Attention

Questions ?