Synchoricity as the basis for going
Beyond Moore

Ahmed Hemani
Professor, Dept. Of Electronics, School of EECS, KTH, Stockholm Sweden
Email: hemani@kth.se
Going Beyond Moore!

**Solutions to go beyond Moore**

1. **Squeeze more out of CMOS**
   a. ASICs like custom functional hardware
   b. Delivers 2-4 orders better energy-delay product compared to GPUs, FPGAs and Multi-cores

2. **Complement CMOS with emerging technologies**
   a. 2.5D and 3D Integration (DRAM)
   b. Computation in memory using Memristors
   c. Plasmonics

```
“Science makes progress, not when you find a solution, but when you make it easy to use the solution”

--- Venki Ramakrishnan, Nobel Laureate
```
**Synchronicity**

*Time is discretized using clock ticks*

Can be *temporally* composed if

\[ \text{clk}_1 = \text{clk}_2 \]

&

The two clocks are skew aligned

---

**Synchoricity**

*Space is discretized using a virtual grid*

Can be *spatially* composed if

If the number of grid cells in each dimension are equal

&

Their interconnect edges are abutable
SiLago (Silicon Lego) Blocks

SiLago Blocks are the new standard cells

- RTL & Coarse Grain Reconfigurable
- 4-5 orders larger than Standard Cells
- Characterized with postlayout data
- Empowers Synthesis from Higher Abstractions
- Inter SiLago Block Wires bought to periphery at right place and right metal layer to enable composition by abutment

(c) Ahmed Hemani
VLSI Designs are Composed by Abutting SiLago Blocks

All Wires – functional and infrastructural (reset, clocks and power grid) are created as a result of abutment

Cost-Metrics of the composite design becomes known with post layout accuracy

(c) Ahmed Hemani
Inspiration from Construction Industry
An Analogy

(c) Ahmed Hemani
We shifted to pre-fabricated wall segments

1. Productivity gain did not *solely* come from the large size of the pre-fabricated wall segments

2. Productivity gain came from physical design discipline that enables composition by abutment

3. IPs in VLSI Design lack this discipline and composition by abutment
### The Berkeley Dwarfs

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dense Linear Algebra</td>
</tr>
<tr>
<td>2</td>
<td>Sparse Linear Algebra</td>
</tr>
<tr>
<td>3</td>
<td>Spectral Methods</td>
</tr>
<tr>
<td>4</td>
<td>N Body Methods</td>
</tr>
<tr>
<td>5</td>
<td>Structured Grids</td>
</tr>
<tr>
<td>6</td>
<td>Unstructured Grids</td>
</tr>
<tr>
<td>7</td>
<td>MapReduce</td>
</tr>
<tr>
<td>8</td>
<td>Combinational Logic</td>
</tr>
<tr>
<td>9</td>
<td>Graph Traversal</td>
</tr>
<tr>
<td>10</td>
<td>Dynamic Programming</td>
</tr>
<tr>
<td>11</td>
<td>Back-track and Branch &amp; Bound</td>
</tr>
<tr>
<td>12</td>
<td>Graphical Models</td>
</tr>
<tr>
<td>13</td>
<td>Finite State Machine</td>
</tr>
</tbody>
</table>

### Region Types – SiLago Block Types

#### Functional
- Graph Theory
- Outer Modem
- Inner Modem
- Protocol Processing
- Spectral Methods
- Dense Linear Algebra
- Sparse Linear Algebra
- Dynamic Programming
- State Machines

#### Infrastructural
- NOCs
- Scratch Pad Memory
- PLL + CGU
- Power Management
- Memory Controller
- FIFO, FIFO Controller
- RISC Processors – RISC-V
- DMA
- Memory Consistency

© Ahmed Hemani
Hardware Centric vs. Software Centric Accelerators vs. Flexilators

Software Centric Platform Based Design

- By default, functionalities are mapped as software.
- Only power and performance critical functionalities are mapped as hardware accelerators.

Hardware Centric Synchoros VLSI Design

- By default, functionalities are mapped as custom functional hardware.
- Only flexibility critical, dynamic and non-deterministic functionalities are mapped to SiLago Flexilators: RISC-V, FSMs, FIFOs, Arbiters, Schedulers, NOCs etc.

Lego Flexilators
Why does Synchoros VLSI Design Work?

Full Custom Mead-Conway
$O(10K \text{ Gates})$

Standard Cells
$O(10 \text{ million Gates})$

Synchoros VLSI Design Style
$O(100 \text{ million Gates})$

- System
- Application
- Algorithm
- RTL
- Boolean
- Std. Cells
- Physical
- GDSII

Log (# of Solutions)

Function Verification (FV)
Constraints Verification (CV)
Manual
Automated

Hardware-Centric Functional ASICs
HPC LIB Impl

Software-Centric Infrastructural SOCs

~300 MUSD

One Time Engineering

~118

© Ahmed Hemani
SiLago Application Level Synthesis

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

HPC Application Algorithms

Sampling Rate, Total Latency

Number and types of SiLago blocks + Mapping

Compose Ready-to-manufacture Chip

Mask Patterns

Reports

(c) Ahmed Hemani
SiLago Design Instances = $\Sigma$ Region Instances

Type, Number, Size and Position of SiLago Region Instances Decided by Synthesis Tools Based on Functionality And Constraints
SiLago can also potentially reduce the manufacturing cost

All SiLago designs are composed of a finite number of SiLago block Types

All SiLago blocks can only have a finite types of neighbors

Each SiLago blocks’s mask depending on the neighbor types can be saved as a component mask

The entire design mask can be composed from such component masks

The DFT Cost can also be factored out

The DFT can be made much more efficient reducing time spent on ATE
What becomes possible

Going Beyond Moore!

**Solutions to go beyond Moore**

1. Squeeze more out of CMOS
   a. ASICs like custom functional hardware
   b. Delivers 2-4 orders better energy-delay product compared to GPUs, FPGAs and Multi-cores

2. Complement CMOS with emerging technologies
   a. 2.5D and 3D Integration (DRAM)
   b. Computation in memory using Memristors
   c. Plasmonics

“Science makes progress, not when you find a solution, but when you make it easy to use the solution”

-- Venki Ramakrishnan
BCPNN
Bayesian Confidence Propagation Neural Network
Professor Anders Lansner
BCPNN Requirements

Functional Requirements: Human Scale - Realtime

1. Realtime simulation
2. 2 Million HCUs – non-deterministically concurrent
3. 170 TFlops/s – BCPNN Computation
4. 50 TBs – Synaptic Weight Storage
5. 200 TBs / s – Bandwidth for synaptic storage
6. 250 GBs / s – Spiking Bandwidth

Infrastructural Requirements
The BCPNN Computation Model

Human Scale Cortex Dimensions

HCU = \sum \text{MCUs}

MCU State Vector

Column updates are more expensive

Input Spike Computation

Output Spike Computation

Support Computation 100 / s

Delay Buffer 100 \times 100 Spikes/s

10 000 Spikes/s

High Level of Temporal Locality

10 000 Connections

100 MCUs

100 MCUs

HCU

MCU Row

Synaptic Memory (25 MB)
Infrastructural Operations are Significant

Infrastructural Operations

170 TFlops

Incoming Spikes Queue & Controller

Input Computation Controller

Scratchpad Memories

Input Computation

Unit R₁ SP FPUs

Input Computation FSM

Output Computation Controller

Scratchpad Memories

Output Computation

Unit R₂ SP FPUs

Output Computation FSM

HCU State Storage Memory Interface

Delay Buffers & Controller for fanout spikes

Outgoing Spikes Queue & Controller

ms Timer

Incoming Spike Distribution Interconnect

ISDN

outSGN

© Ahmed Hemani
The Silicon Lego Bricks for Method Applied to BCPNN
A Structured Physical Design Scheme to enable System-level synthesis
**BCPNN Implementation**

**H-Cube**
- 8 layers of DRAM
- 1 Bank per layer
- 2 Banks / HCU
- TSV Micro Channel
- 4 HCUs + Control

**BCU: Brain Computation Unit**
- 1.529 X 1.729 mm²
- 32 H-Cubes
- 32 Micro Channels

In Collaboration with Prof. Nobert When and Dr. Christian Wiess
TU Kaiserslautern
**BCPNN: ASIC vs GPUs**

(a) Energy Breakdown ASIC

- Infrastructure: 3%
- SRAM: 9%
- DRAM: 73%
- Computation: 15%

Energy Delay Product: 3.06 kJ · s

(b) Energy Breakdown GPUs

- Infrastructure: 20%
- SRAM: 9%
- DRAM: 60%
- Computation: 20%

ASiC: 3.0 kW

GPUs: 2.6 MW

SpiNNaker-2 comparable to GPUs

Energy Delay Product: 2642 kJ · s

# of GK210 cores: 5000
Energy: 563.1 kJ
1s Realtime: 4.69 s simulated time

© Ahmed Hemani
The Impact of Column Access Elimination + Exploiting Temporal Locality

Baseline

Column Access Eliminated

Column Access Eliminated + Temporal Locality

Such optimizations can be automatically inferred from Simulations
Interconnect and Storage are Expensive

3.2 pJ = 32 bit Data 1 mm ≈ 32-bit FLOP > accessing 1 bit in 3D integrated DRAM

Computation in Memory using Memristors

Benefits:
1. Single cycle dot product
2. Can be extended to do addition, multiplication, element wise multiplication, matrix inversion
3. No need to fetch, decode and execute instructions → addresses wire problem
4. In some application instances, initialization of matrix would be a one-time event

Challenges
1. Large matrices will need to be fragmented resulting in movement of data. Need complimentary control circuitry
2. ADC’s consume significant power and inject latency
3. Accuracy
4. Experimental solutions reported. Not part of mainstream design flow

Source of Diagram Above: Chenchen Liu, Qing Yang, Bonan Yan, Xiaocong Du, Hai (Helen) Li, “A Memristor Crossbar Based Computing Engine Optimized for High Speed and Accuracy”, ISVLSI 2016
**Memristor based CIM in the SiLago Framework**

### Region Types – SiLago Block Types

#### Functional
- Graph Theory
- Outer Modem
- Inner Modem
- Protocol Processing
- Spectral Methods
- Dense Linear Algebra
- Sparse Linear Algebra
- Dynamic Programming
- State Machines
- Memristor CIM

#### Infrastructural
- NOCs
- Scratch Pad Memory
- PLL + CGU
- Power Management
- Memory Controller
- FIFO, FIFO Controller
- *RISC Processors – RISC-V*
- NVM
- DRAM Vaults

1. A Memristor CIM in a range of dimensions
2. Characterized with post-layout data and circuit level simulations and validated with test chips
3. Exports, functional matrix operations and infrastructural operations like initializing crossbar, NIU operations, reg file operations etc.
4. Higher abstraction synthesis tools can refine in terms of CIM SiLago blocks and know its performance, energy and area.
25 Watt Biologically Plausible Human Scale Brain

Single Precision Floating Point

Move to 16/32 bit Integer Arithmetic

1. Synaptic Storage/Access will reduce by ~50%
2. Computation Energy will reduce by ~75%

~800 Watts → ~250 Watts

ReRAM Computation in Memory

~25 Watts

Caveat:
Based on best effort estimates and not on actual implementation

~ 2 TOPs/watt
28 nm bulk CMOS
Wave Based Computing using Plasmons

1. Logic values encoded as phase of the waves
2. Interference of waves interpreted as majority gate computation

In wave computing, information is coded in the phase or the amplitude of the wave.
Plasmonics + CMOS Computing using SiLago blocks

Plasmon detector

CMOS Logic Drivers

Plasmon Waveguides

Electrally driven Plasmon sources

Plasmon Sources

Phase Modulators

THz

Plasmon Logic Circuit

High Speed

Demodulator

Output

SiLago Interconnect Waveguide

SiLago micro-architecture block (full adder)

Plasmon Exa – Plasmonics based Exa-scale computing design. Synthesized in terms of Silicon Lego (SiLago) blocks.
Impact

1. 1000 X Power Density
2. More Affordable

Software Centric / GPU + Based Computing

Hardware Centric SiLago Based Computing