**MBARC: A Scalable Memory Based Reconfigurable Computing Framework for Nanoscale Devices**

Somnath Paul and Swarup Bhunia
Department of EECS, Case Western Reserve University

Presented by Prof. Kaushik Roy
Department of ECE, Purdue University
Outline

- Introduction
  - Background
  - Motivation
- Memory based computing
- Design considerations
  - Memory array implementation with nano-devices
  - Scalability
- Test Setup & Results
- Conclusion
Introduction

Background

• Quest for potential alternative to CMOS at the end of its roadmap

• Emerging novel devices include: Single Electron Transistor (SET), Quantum-dot Cellular Automata (QCA), Carbon Nanotube FET (CNTFET) and Chemically Assembled Electronic Nano-computers (CAEN) etc.

• Molecular electronics hold tremendous potential in terms of:
  
  i) High integration density (~$10^{10}$ devices/cm$^2$)
  
  ii) Low power operation
  
  iii) High switching speed
Introduction

Background

• Nanoscale molecular circuits comprise of a monolayer of rotaxane molecule sandwiched between Ti/Pt metal nano-wires

• Experimental success in realizing molecular crossbar circuits

• Substantial progress in the development of
  
  i. Architecture

  ii. Circuit

  iii. Efficient CAD tools for application mapping and testing of the molecular crossbars.

• Each crossbar junction acts as 1-bit storage element

• Highly amenable for a dense and regular reconfigurable fabric
**Introduction**

**Motivation**

- Bistable rotaxane molecules modeled as diode-like devices
- Absence of signal restoration
- Requirement for interfacing the nano-crossbar with conventional CMOS devices
- Mismatch in pitch dimensions between nano-wire (~nm) and CMOS interconnect (~µm)
- Essential to choose an architectural framework that
  1. Preserves the high integration density of nano-crossbars
  2. Tolerates high defect rate for the molecular devices
Memory Based Computation

Main Idea

- Decompose a logic circuit into a set of partitions
- Implement the partitions as LUTs in a nanoscale memory array
- Use a CMOS based controller to evaluate the partitions in topological and time-multiplexed manner

Primary Components

- Partitions are mapped to different memory modules, collectively referred to as function table
- Information regarding address, schedule and connectivity of the partitions stored in the schedule table
- Schedule table, function table and intermediate registers are collectively known as MCB (Memory-based Computational Block)
Memory Based Computation Scheme

Design flow for MBARC

Inputs:
- Gate Level Netlist;
- Max I/O for a partition

1. Topologically sort the netlist nodes
2. Read design constraints

Partitioning Step

- Memory-Aware Partitioning
- PLP-Aware Partitioning

Partitioned Netlist

Scheduling Algorithm

Schedule the Partitions

Map to Function Table and Schedule Table

Outputs:
1. Schedule Table with Partition Sequence and Partition Address,
2. Function Table with Function Responses

ASP-DAC 22nd Jan, 2008
Memory Based Computation

- **Software architecture**
  - Heuristic based solutions for Memory and PLP (Partition Level Parallelism) aware partitioning of the target application
  - Static scheduling of the partitions during the compilation phase

- **Advantages**
  - CMOS interfacing logic in MBARC is localized and separate, potentially facilitating CMOS-nano hybridization process
  - Existing techniques to test, diagnose and achieve defect tolerance in memory can be used for the proposed framework
Memory array implementation with nano-devices

- Emerging nano-devices are typically dense and periodic structures amenable to large memory array design
- A monolithic large defect-free memory block design is challenging for these devices
- Desirable to design a larger memory from smaller nano-crossbars with shorter access times

Organization of a large memory module from 4 individual NXN memory blocks
Scalability

- **MBARC** framework extended to exploit parallelism among partitions – Exploiting thread-level parallelism
- Logic for most applications are bit-sliced
- Identifying the parallel threads in the design prior to the partitioning procedure
- Reduces the number of partitions in the critical path, thus reducing the total evaluation time

Framework for realization of thread level parallelism and pipelining
Design Considerations

- **Scalability**
  - Proposed framework inherently supports multi-cycle evaluation
  - Each MCB can be pipelined with another to allow pipelined design for improved throughput

- **Different granularities of computation**
  - Design can be evaluated at a single MCB
  - Design can be evaluated in a single pipelined thread
  - Design can be decomposed into several parallel threads

- **A hybrid reconfigurable platform with both time and space multiplexing results in significant performance improvement**
Test Setup

- The proposed framework was validated for ISCAS benchmark circuits
- Design overhead for MBARC compared against a Nano-FPGA* (nano-crossbar based FPGA) model
- Methodology
  - Benchmarks synthesized and technology mapped using RASP
  - Synthesized netlist partitioned into multi-input multi-output partitions
  - To evaluate a N×M partition, a total of $2N \times 2N + 2N \times M$ data points are required in nano-crossbar

## Results for partitioning algorithms

<table>
<thead>
<tr>
<th>ISCAS 85 Ckt</th>
<th>Memory Aware Partitioning</th>
<th>PLP Aware Partitioning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12 X 12 X 4</td>
<td>12 X 12 X 8</td>
</tr>
<tr>
<td></td>
<td>Mem Req (kB)</td>
<td>Delay (cyc)</td>
</tr>
<tr>
<td>C1908</td>
<td>18.4</td>
<td>8</td>
</tr>
<tr>
<td>C1355</td>
<td>19.8</td>
<td>7</td>
</tr>
<tr>
<td>C2670</td>
<td>72.9</td>
<td>12</td>
</tr>
<tr>
<td>C3540</td>
<td>61.8</td>
<td>19</td>
</tr>
<tr>
<td>C5315</td>
<td>89.4</td>
<td>19</td>
</tr>
<tr>
<td>C7552</td>
<td>108.2</td>
<td>28</td>
</tr>
<tr>
<td>C6288</td>
<td>78.4</td>
<td>19</td>
</tr>
</tbody>
</table>
Results

- Percentage improvement in design overhead for 12 input 12 output partitions
- 8 ported memory to support 8 parallel partition evaluation in MBARC
- Number of LUT and programmable interconnects for Nano-FPGA estimated using Quartus V7.0 for Stratix III FPGA platform
- MBARC allows 5%, 36.3% and 31.6% average savings in area, delay and energy/vector compared to Nano-FPGA model
## Results

### Design overhead for thread level parallelism

<table>
<thead>
<tr>
<th>Benchmark Ckts</th>
<th>MBARC (12X12X4)</th>
<th>NanoFPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of threads</td>
<td>Delay (ns)</td>
</tr>
<tr>
<td>C5315</td>
<td>7</td>
<td>6.5</td>
</tr>
<tr>
<td>S13207</td>
<td>8</td>
<td>10.4</td>
</tr>
<tr>
<td>S15850</td>
<td>8</td>
<td>9.1</td>
</tr>
</tbody>
</table>

For a 4-port memory configuration, thread level parallel evaluation allows 27.9% and 32.7% improvement in total execution time and area respectively at the cost of 25.3% increase in the total energy/vector. 
Conclusion

- **MBARC** is a reconfigurable memory-based computing model for emerging nanoscale devices
- Proposed model evaluates a logic function in time multiplexed manner
- It minimizes the requirement for programmable interconnect as well as interfacing hardware
- The framework can be easily scaled to exploit parallel execution of multiple partitions and threads
- Investigations reveal that a nano-crossbar based MBARC framework offers significant improvements in area, power and performance compared to a FPGA-like purely spatial framework
Thank You