Architecture

Philip Leong
Computer Engineering Laboratory
School of Electrical and Information Engineering
The University of Sydney
1. Introduction to Reconfigurable Computing
   - what is reconfigurable computing, applications, areas for research
2. Abstractions for Implementation
   - microcode, map-reduce
3. Monte Carlo Simulation
   - uniform and Gaussian random number generation, Monte Carlo arithmetic
4. FPGA Architecture
   - floating-point FPGAs, process variation
Describe how FPGAs can be

- Specialised for Floating-point applications
- Used as a technique to address process variations
Describe how FPGAs can be

- Specialised for Floating-point applications
- Used as a technique to address process variations
Floating-point FPGAs

1. Introduction
2. FPGA with Embedded FPU
3. Hybrid FPGA: Architecture
4. Conclusion
Introduction

› Fixed point computation dominant in FPGA designs due to speed and efficiency
› Floating point FPGAs suitable for DSP and HPC applications
› Increased parallelism and higher memory bandwidth
› What modifications are required to FPGA architectures for floating point?
Key results

› VEB method for studying embedded blocks
› Domain-specific hybrid FPGA architecture
   - reconfigurable resources: multiple granularity
   - customised for different applications
   - model based on current FPGAs and CAD tools
› Hybrid FPGA for floating point applications
   - parameterised coarse-grained block
   - 6 Benchmarks, compared with Virtex-II device
   - 18 times area reduction, 2.5 times speedup
Computational Density

Normalized Performance Comparison

- FPGA
- Processors SPECfp 2000
- Processor Peak

January 12, 2004

BWRC, UC Berkeley
FPGA Strengths
- More parallelism
- Higher computational density
- Lower power consumption
- Higher memory bandwidth, direct control of accesses
- Can be fault tolerant

Weaknesses
- Long wordlengths
- Floating point
- Low clock frequency
- Run out of resources
- Design time
- Legacy code
Floating point FPGAs

› Commercial FPGA Weaknesses
  - Long wordlengths
  - Low clock frequency
  - Run out of resources
  - Design time

› Domain-specific FPGA for FP
  - Hardwired FPUs
  - Reconfigurable fabric
  - Dynamic Reconfiguration

- Advantages
  □ More transistors for computing than a uP
  □ Better FP perf than FPGA/uP
  □ Development time reduced as designers do not need to deal with fixed point quantisation issues
  □ External memory often bottleneck, FPGAs offer potentially higher bandwidth (multiple channels) as well as custom control of cache
  □ Branch mispredictions don’t cause tens of cycles to recover
Floating-point FPGAs

1. Introduction

2. FPGA with Embedded FPU

3. Hybrid FPGA

4. Conclusion
Virtual Embedded Blocks

› Use existing tools to study effects of embedded elements in FPGAs

› Use commercial design flows
  - Closer to real architectures
  - Optimisations such as retiming
  - Greatly reduced development time

› Explore technology trends based on systematic variation of VEB parameters in applications.
Virtual Embedded Blocks: VEB

- Dummy blocks
  - model coarse-grained unit’s area and delay
- Timing analyser
  - estimate performance, e.g. fine-to-coarse delay
Area Model

- Use logic cells to model real ASIC embedded blocks
- Estimate LC area (normalised to feature size) from die photos
### Area Model

- Estimates of logic cell area including configuration bit, buffer and interconnect overheads.
- Area based on estimate that 70% of the total die area for logic cells, the other area being for pads, block memories, multipliers etc.
- Normalised to feature size

<table>
<thead>
<tr>
<th>Device</th>
<th>LCs/CLB $L$</th>
<th>Area/CLB $A$ ($\mu m^2$)</th>
<th>Feature Size $f$ ($\mu m$)</th>
<th>Normalised LC area $(N = A / Lf^2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apex 20K400E [8]</td>
<td>10</td>
<td>63161</td>
<td>0.18</td>
<td>195,000</td>
</tr>
<tr>
<td>Virtex E [8]</td>
<td>4</td>
<td>35462</td>
<td>0.18</td>
<td>267,000</td>
</tr>
<tr>
<td>Virtex II 3000 [9]</td>
<td>8</td>
<td>$71,429 \times 0.7$</td>
<td>0.12</td>
<td>434,000</td>
</tr>
<tr>
<td>Virtex II 1000 [10]</td>
<td>8</td>
<td>$72,782 \times 0.7$</td>
<td>0.12</td>
<td>442,000</td>
</tr>
</tbody>
</table>
Area model can be used to compare logic cell area to any embedded block

- Logic Cell (LC): 442,000 = 1 LC
- Multiplier: 2,751,000 (normalised) ~ 6 LC
- Blue-gene floating point unit (FPU) ~ 570 LC
› Match delays using LC
  - Use adder carry chain to model the delay
› For small blocks, may fail to match both area and delay
Benchmarks

› Large unsigned multiplier
  - Formed by cascading embedded block multiplier
  - 34-bit (mul34), 68-bit (mul68), 136-bit (mul136)

› Brace, Gatarek and Musiela (BGM) framework
  - Price interest rate derivative using Monte Carlo simulation
  - A good example for retiming because of its complexity

› Fly compiler
  - Allow a single description to generate a circuit with different arithmetic operations
  - Applications include
    - Digital sine cosine generator (dscg)
    - ordinary differential equation solver (ode)
    - 3x3 matrix multiplication (mm3)
    - Finite impulse response filter (fir4)
    - Butterfly stage for discrete Fourier transform (bfly)

› 18-bit fixed point and double precision floating point circuits generated
## Verification using Embedded Multipliers

<table>
<thead>
<tr>
<th></th>
<th>EM delay (ns)</th>
<th>VEB delay (ns)</th>
<th>Overall Diff (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSCG</td>
<td>4.599</td>
<td>4.981</td>
<td>8</td>
</tr>
<tr>
<td>FIR4</td>
<td>4.616</td>
<td>4.794</td>
<td>2</td>
</tr>
<tr>
<td>ODE</td>
<td>4.402</td>
<td>4.539</td>
<td>3</td>
</tr>
<tr>
<td>MM3</td>
<td>4.859</td>
<td>4.815</td>
<td>1</td>
</tr>
<tr>
<td>BFLY</td>
<td>5.668</td>
<td>5.224</td>
<td>8</td>
</tr>
<tr>
<td>MUL34</td>
<td>11.191</td>
<td>11.287</td>
<td>1</td>
</tr>
<tr>
<td>MUL68</td>
<td>12.553</td>
<td>14.099</td>
<td>11</td>
</tr>
<tr>
<td>MUL136</td>
<td>14.632</td>
<td>13.248</td>
<td>10</td>
</tr>
<tr>
<td>BGM</td>
<td>14.055</td>
<td>13.866</td>
<td>1</td>
</tr>
<tr>
<td>BGM (retimed)</td>
<td>11.594</td>
<td>11.602</td>
<td>0</td>
</tr>
</tbody>
</table>
Effect of Faster Multipliers (BGM)

![Graph showing the effect of faster multipliers on system performance. The x-axis represents embedded multiplier performance, and the y-axis represents system performance. The graph shows an increasing trend as the multiplier performance increases.](image-url)
Embedding a floating point unit
- FPU delay and area based on published Blue Gene data
  - 700MHz, 4.26 $mm^2 = 570$ LCs
- For FPGAs, reduce latency and clock frequency by a factor of 5: 140 MHz, one cycle latency

Explore the speedup by increasing the performance of floating point unit
- Tested on floating-point butterfly circuit (bfly)
# Effect of FPU on System Performance

<table>
<thead>
<tr>
<th></th>
<th>FPGA</th>
<th>VEB</th>
<th>Reduction Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>size (LC)</td>
<td>delay (ns)</td>
<td>size (LC)</td>
</tr>
<tr>
<td>dscg</td>
<td>19006</td>
<td>22.711</td>
<td>3420 + 940</td>
</tr>
<tr>
<td>fir4</td>
<td>20590</td>
<td>23.545</td>
<td>3990 + 996</td>
</tr>
<tr>
<td>ode</td>
<td>13984</td>
<td>17.756</td>
<td>2850 + 870</td>
</tr>
<tr>
<td>mm3</td>
<td>17236</td>
<td>19.320</td>
<td>2850 + 2390</td>
</tr>
<tr>
<td>bfly</td>
<td>25640</td>
<td>20.245</td>
<td>4560 + 3424</td>
</tr>
<tr>
<td></td>
<td>Geometric Mean:</td>
<td>3.7</td>
<td>4.4</td>
</tr>
</tbody>
</table>
Observations

› Embedded FPUs: 4x speed and density
› Retiming important (20% diff in speed)
› Inefficiencies when 1-bit LUTs used to process 64-bit datapaths
› Coarser-grained blocks may offer higher performance
1. Introduction

2. FPGA with Embedded FPU

3. Hybrid FPGA

4. Conclusion
Domain-specific Hybrid FPGAs

- Reconfigurable resources of multiple granularity
- Parameterised
- Domain-specific
› Most digital circuits
  - datapath: regular, word-based logic
  - control: irregular, bit-based logic

› Hybrid FPGA
  - customised coarse-grained block: domain-specific requirements
  - fine-grained blocks: existing FPGA architecture
  - good match to computing applications for a given domain
Analysis

› Modelling
  - synthesizable coarse-grained fabric (VEB)

› Evaluation
  - place & route benchmarks on hybrid FPGA
  - measure speed and area

› Exploration
  - cover large design space of architectures
Coarse-grained units
- dominated by multiplication and addition
- IEEE double precision floating point
- 64-bit datapaths much more efficient: shared routing and configuration, specialised logic

Fine-grained units
- implement control logic, state machine
- based on Xilinx Virtex-II
Coarse-grained fabric design

› Coarse-grained block synthesized
  - ASIC delay/size parameters from standard cells
› HDL allows parameterisation of architecture
  - number of embedded floating point operators
  - number of feedback registers, etc
Coarse-grained fabric

D=9, M=4, R=3, F=3, 2 add, 2 mul: best density over benchmarks
Evaluation

- 6 benchmark circuits
  - DSP computation kernels: e.g. bfly
  - linear algebra: e.g. matrix multiplication
  - complete application: e.g. bgm
- Circuits: partitioned to control + datapath
  - control: vendor tools to fine-grained units
  - datapath: manually map to coarse-grained units
- Comparison
  - directly synthesized to Xilinx Virtex-II devices
Example floorplan: bgm
## Floating Point hybrid FPGA

<table>
<thead>
<tr>
<th>Function</th>
<th>Area (slices)</th>
<th>Delay (ns)</th>
<th>Area (slices)</th>
<th>Delay (ns)</th>
<th>Area (times)</th>
<th>Delay (times)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bfly</td>
<td>565</td>
<td>9.02</td>
<td>13733</td>
<td>24.57</td>
<td>24.3</td>
<td>2.72</td>
</tr>
<tr>
<td>dscg</td>
<td>661</td>
<td>10.11</td>
<td>9614</td>
<td>22.78</td>
<td>14.5</td>
<td>2.25</td>
</tr>
<tr>
<td>fir4</td>
<td>371</td>
<td>9.06</td>
<td>11290</td>
<td>23.68</td>
<td>30.4</td>
<td>2.61</td>
</tr>
<tr>
<td>mm3</td>
<td>642</td>
<td>8.90</td>
<td>8889</td>
<td>23.4</td>
<td>13.8</td>
<td>2.63</td>
</tr>
<tr>
<td>ode</td>
<td>545</td>
<td>9.74</td>
<td>8238</td>
<td>21.93</td>
<td>15.1</td>
<td>2.25</td>
</tr>
<tr>
<td>bgm</td>
<td>1810</td>
<td>10.00</td>
<td>30207</td>
<td>24.34</td>
<td>16.7</td>
<td>2.43</td>
</tr>
</tbody>
</table>

**Geometric Mean**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>Delay</td>
</tr>
<tr>
<td>18.3</td>
<td>2.48</td>
</tr>
</tbody>
</table>
Conclusion

› Domain-specific hybrid FPGAs
  - fine-grained + synthesizable coarse-grained units
  - explore customisations using existing FPGA tools

› Domain-specific floating point FPGA
  - 18 times area reduction
  - 2.5 times speedup
  - greatly improved area over island-style+FPU

\[ \text{compared with Virtex-II} \]
Future work

› Estimating routing overhead
› Explore different coarse-grained units for
  - floating point operations
› Automated design tools
  - mapping to coarse-grained units
  - partitioning between coarse and fine-grained units
› Other domain-specific applications
  - scientific computing
  - imaging
  - networking
Describe how FPGAs can be

- Specialised for Floating-point applications
- Used as a technique to address process variations
› **Process variation** increase as the transistor feature size reduces due to the difficulty of fabricating small structures consistently across a die or a wafer.

› These variations cause mismatches between identical structures.
  - **Timing**
  - Leakage
  - Dynamic Power
Sources of Variations

Random dopant fluctuation (RDF)

Gate oxide thickness

Line-edge roughness (LER)
Line-width roughness (LWR)

Optical proximity correction (OPC)

Unique variation distribution patterns can be used to create chip identifiers (chip-IDs).

Fine-grained characterization of process variation is required for VAD.

A more efficient VAD method without need for characterization is proposed.

FPGA-Based
Develop fine-grained process variation characterization techniques to facilitate variation aware design (VAD).
• Process variation is becoming increasingly severe.

• Post-silicon optimization according to variation distribution

• **NO** practical process variation characterization

Distribution of logic element delay
Targets for Characterization

› Homogeneous “island-style”
  - Logic Block
  - Interconnection
  - I/O Cells

› Xilinx Spartan-3e (90nm)
  - Configurable logic block (CLB)
    - CLB: 8 logic elements (LEs)
    - LE (4-input look-up table or LUT)
  - Switch Matrix (SM)
  - Global routing:
    - direct connection
    - double wire
    - hex wire
$D_{\text{loop}} = \frac{T}{2 \times C}$

$T$ denotes the time interval for RO to run, which is set by a timer, $C$ denotes the counter values recorded during $T$. 
To minimize $\Delta D_{\text{int}}$

CONSIDER bounce

$$\Delta D_{\text{Loop}} = D_{\text{LE1}} + \Delta D_{\text{int}}$$
The table summarizes contribution of interconnect delay for ROs (both for 8-stage and 7-stage).

<table>
<thead>
<tr>
<th>Composition</th>
<th>Est. $D_{\text{loop}}$ (ns)</th>
<th>Est. $D_{\text{int}}$ (ns)</th>
<th>% of $D_{\text{int}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,3,2,6,4,5,7,8</td>
<td>6.262</td>
<td>0.182</td>
<td>2.91%</td>
</tr>
<tr>
<td>2,6,4,5,7,8,3</td>
<td>5.478</td>
<td>0.158</td>
<td>2.88%</td>
</tr>
<tr>
<td>1,3,6,4,5,7,8</td>
<td>5.528</td>
<td>0.208</td>
<td>3.76%</td>
</tr>
<tr>
<td>1,2,6,4,5,7,8</td>
<td>5.478</td>
<td>0.158</td>
<td>2.88%</td>
</tr>
<tr>
<td>1,3,2,6,5,7,8</td>
<td>5.568</td>
<td>0.248</td>
<td>4.45%</td>
</tr>
<tr>
<td>1,3,2,6,4,7,8</td>
<td>5.481</td>
<td>0.161</td>
<td>2.94%</td>
</tr>
<tr>
<td>1,2,5,3,4,7,8</td>
<td>5.595</td>
<td>0.275</td>
<td>4.92%</td>
</tr>
<tr>
<td>1,3,2,6,4,5,8</td>
<td>5.481</td>
<td>0.161</td>
<td>2.94%</td>
</tr>
<tr>
<td>1,3,2,5,7,6,4</td>
<td>5.595</td>
<td>0.275</td>
<td>4.92%</td>
</tr>
</tbody>
</table>
Interconnect delay can be calculated by subtracting delay of LEs (available after LE delay characterization) from the RO loop delay.

We characterize delay of a pair of interconnects, not explicitly a wire segment.

We can also estimate delay of a mix of these …
Interconnect Delay Variation Estimation

- **Coverage Rate**
  \[ R_C = \sum_{i=1}^{n} \frac{D_{O_i}}{D_{path}} \]
  overlapped part
  target path delay

- **Contribution Factor**
  \[ F_C = \frac{D_O}{D_{int}} \]
  interconnect delay

---
uncovered

covered
Experimental Results

Scaling Factor

\[ F_s = \frac{D_{\text{real}}}{D_{\text{est}}} \]

- \( D_{\text{real}} \): Measured delay
- \( D_{\text{est}} \): Delay estimated by timing tool

<table>
<thead>
<tr>
<th>RO Types</th>
<th>( D_{\text{est}} )</th>
<th>( D_{\text{real}} )</th>
<th>( F_s )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-stage</td>
<td>3.162</td>
<td>1.767</td>
<td>0.559</td>
</tr>
<tr>
<td>5-stage</td>
<td>3.915</td>
<td>2.148</td>
<td>0.549</td>
</tr>
<tr>
<td>6-stage</td>
<td>4.697</td>
<td>2.574</td>
<td>0.548</td>
</tr>
<tr>
<td>7-stage</td>
<td>5.481</td>
<td>3.028</td>
<td>0.552</td>
</tr>
<tr>
<td>8-stage</td>
<td>6.262</td>
<td>3.424</td>
<td>0.546</td>
</tr>
</tbody>
</table>

The margin is for \textbf{process}, \textbf{voltage} and \textbf{temperature} (PVT) variation tolerance.
### Measured LE & Interconnect Delay

<table>
<thead>
<tr>
<th>LE#</th>
<th>mean (ns)</th>
<th>% of 3-sigma</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.383</td>
<td>11.50%</td>
</tr>
<tr>
<td>2</td>
<td>0.403</td>
<td>14.70%</td>
</tr>
<tr>
<td>3</td>
<td>0.421</td>
<td>12.30%</td>
</tr>
<tr>
<td>4</td>
<td>0.407</td>
<td>11.30%</td>
</tr>
<tr>
<td>5</td>
<td>0.486</td>
<td>15.60%</td>
</tr>
<tr>
<td>6</td>
<td>0.454</td>
<td>14.20%</td>
</tr>
<tr>
<td>7</td>
<td>0.458</td>
<td>12.60%</td>
</tr>
<tr>
<td>8</td>
<td>0.465</td>
<td>14.10%</td>
</tr>
</tbody>
</table>

LE 5-8 can be configured as SRAM.

We take half of connection pair delay for result presentation, which are **233.1ps**, **271.3ps** and **358.6ps** for connections with direct connection, double and hex wire respectively. 3-sigma width is about 10% of the mean value.
Taking LE 1 as the object of comparison, chip #1 is faster than chip #2 by 7.6% on average.
Tested 5 different ROs with different amounts of interconnect.

<table>
<thead>
<tr>
<th>Case #</th>
<th>% of $D_{LE}$</th>
<th>% of $D_{int}$</th>
<th>$D_{diff,meas}$ (ps)</th>
<th>$D_{diff,est}$ (ps)</th>
<th>$R_{err}$ %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>68.90%</td>
<td>31.10%</td>
<td>55.8</td>
<td>62.0</td>
<td>11.10%</td>
</tr>
<tr>
<td>2</td>
<td>55.70%</td>
<td>44.30%</td>
<td>76.0</td>
<td>79.1</td>
<td>4.08%</td>
</tr>
<tr>
<td>3</td>
<td>49.50%</td>
<td>50.50%</td>
<td>86.0</td>
<td>93.8</td>
<td>9.20%</td>
</tr>
<tr>
<td>4</td>
<td>43.00%</td>
<td>57.00%</td>
<td>41.5</td>
<td>38.9</td>
<td>6.27%</td>
</tr>
<tr>
<td>5</td>
<td>35.70%</td>
<td>64.30%</td>
<td>52.6</td>
<td>57.2</td>
<td>8.75%</td>
</tr>
</tbody>
</table>
Conclusion

› Contribution

- self-characterization of FPGA using intrinsic resources
- A step towards practical variation aware design methodology
1. Characterization

Process Variation

2. Variation Aware Design

3. Physical Unclonable Function
Unlike ASIC, FPGA design is implemented in post-silicon phase, therefore, several variation-aware design (VAD) methods can be used to improve timing yield.
Variation-aware placement and routing

[Katsuki, 2005] [Srinivasan, 2006] [Cheng, 2006]

- Pros:
  - Effective timing improvement
- Cons:
  - Need for variation characterization
  - Chip-specific optimization is not practical for implementation on a large amount of FPGAs
Mutually exclusive critical path configurations (MECPCs) [Matsumoto, 2007]

- **Pros:**
  - No need for variation characterization
  - Simple and effective

- **Cons:**
  - Still not very efficient
  - CAD quality randomness
  - No awareness of variation in spatial correlation
FPGA’s symmetry provides potential capability to tune the design in post-layout phase.

A combined method with coarse-grained and fine-grained design tuning

- Coarse-grained
  - Configuration rotation and flipping
  - To counter spatial correlation
- Fine-grained
  - Neighboring LEs swap
  - To counter random variation
Configuration Rotation and Flipping

- Homogeneous and Symmetric
- More efficient (Compared to iteratively routing in MECPCs)
Symmetric Architecture

Symmetric CLB
- Even distribution of I/O pins
- Identical CLB ipin to LE pin delay and local interconnect delay

Symmetric Switch Box
- Subset switch box
- Global routing net with a unique track number
One additional multiplexor is added to the VPR FPGA architecture to facilitate path tuning.

**Note:** These multiplexors are already available in Xilinx FPGAs to optimize large multiplexors.
To identify critical path from initial design
To modify critical path with duplications and measure performance
LE Swap – Step 3

To swap LEs for performance measurement

---

Critical Path

Non-Critical Path
Inter-die variation affects all components of a chip identically.

Within-die variation
- Spatial Correlation
  \[ \rho = \begin{cases} 
  1 - \frac{d}{D_L}(1 - \rho_B) & d \leq D_L \\
  \rho_B & d \geq D_L 
  \end{cases} \]
- Random Variation
  \[ \sigma_{V_{th}} = \frac{C}{\sqrt{L_{eff} W_{eff}}} \]

- 16nm predictive technology model (PTM)
This distribution contains 10x10 dies. Both die-to-die (within wafer) and within-die systematic variation are considered using this model.
Experiment

› Setup

- **20 MCNC** benchmark circuits are used to verify our method.
- **1000 FPGA variation distributions** are constructed for simulation.

› Evaluation Metrics

- Given a timing performance target, evaluate the proportion of chips that satisfy the timing constraint.
- Given a yield target, evaluate how fast a design can operate over a number of circuits and meet a yield constraint.
New Design Flow

Original Design Flow

- Synthesis
- Configuration
- Rotation and Flipping
- Placement

Conf. #0
Conf. #1
Conf. #7

- Placement Outputting
- Performance Evaluation
- Best of conf. #0...7
- LE Swap
- Optimized Design

Chip Variation Model
Experimental Results

Yield target: **95% percentile of the distribution**

<table>
<thead>
<tr>
<th></th>
<th>Config. R. &amp; F.</th>
<th>LE Swapping</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Delta_{\mu} %$</td>
<td>$\Delta_{\sigma} %$</td>
<td>$\Delta_{\mu} %$</td>
<td>$\Delta_{\sigma} %$</td>
<td>$\Delta_{D_f} %$</td>
<td>$\Delta_Y %$</td>
</tr>
<tr>
<td>alu4</td>
<td>-10.9</td>
<td>-20.4</td>
<td>-3.81</td>
<td>-0.9</td>
<td>-15.3</td>
<td>89.5</td>
</tr>
<tr>
<td>apex2</td>
<td>-9.99</td>
<td>-13.9</td>
<td>-4.88</td>
<td>-2.4</td>
<td>-14.6</td>
<td>88.9</td>
</tr>
<tr>
<td>apex4</td>
<td>-10.7</td>
<td>-16.9</td>
<td>-2.60</td>
<td>-1.5</td>
<td>-13.8</td>
<td>87.4</td>
</tr>
<tr>
<td>bigkey</td>
<td>-7.07</td>
<td>-10.7</td>
<td>-4.11</td>
<td>0.8</td>
<td>-9.10</td>
<td>74.7</td>
</tr>
<tr>
<td>clma</td>
<td>-8.04</td>
<td>-16.0</td>
<td>-3.68</td>
<td>-3.6</td>
<td>-12.5</td>
<td>89.8</td>
</tr>
<tr>
<td>des</td>
<td>-7.95</td>
<td>-16.0</td>
<td>-5.22</td>
<td>-15.4</td>
<td>-10.6</td>
<td>81.8</td>
</tr>
<tr>
<td>diffeq</td>
<td>-10.6</td>
<td>-16.9</td>
<td>-5.64</td>
<td>-8.0</td>
<td>-16.9</td>
<td>93.7</td>
</tr>
<tr>
<td>dsip</td>
<td>-7.93</td>
<td>-14.7</td>
<td>-4.44</td>
<td>3.4</td>
<td>-10.8</td>
<td>78.5</td>
</tr>
<tr>
<td>elliptic</td>
<td>-8.81</td>
<td>-13.7</td>
<td>-5.54</td>
<td>-12.0</td>
<td>-15.3</td>
<td>94.3</td>
</tr>
<tr>
<td>ex1010</td>
<td>-11.0</td>
<td>-20.5</td>
<td>-2.94</td>
<td>-2.4</td>
<td>-12.6</td>
<td>92.6</td>
</tr>
<tr>
<td>ex5p</td>
<td>-7.21</td>
<td>-8.10</td>
<td>-4.13</td>
<td>1.0</td>
<td>-15.4</td>
<td>78.4</td>
</tr>
<tr>
<td>frisc</td>
<td>-10.6</td>
<td>-18.8</td>
<td>-6.10</td>
<td>-13.1</td>
<td>-16.6</td>
<td>97.9</td>
</tr>
<tr>
<td>misex3</td>
<td>-10.2</td>
<td>-14.4</td>
<td>-3.28</td>
<td>-4.6</td>
<td>-13.9</td>
<td>85.1</td>
</tr>
<tr>
<td>pdc</td>
<td>-6.91</td>
<td>-16.2</td>
<td>-2.95</td>
<td>4.0</td>
<td>-9.85</td>
<td>75.5</td>
</tr>
<tr>
<td>s298</td>
<td>-10.3</td>
<td>-18.2</td>
<td>-4.01</td>
<td>-0.4</td>
<td>-14.6</td>
<td>89.4</td>
</tr>
<tr>
<td>s38417</td>
<td>-5.87</td>
<td>-10.2</td>
<td>-5.51</td>
<td>-8.6</td>
<td>-11.9</td>
<td>89.7</td>
</tr>
<tr>
<td>s38584.1</td>
<td>-8.15</td>
<td>-9.53</td>
<td>-4.84</td>
<td>-10.7</td>
<td>-13.5</td>
<td>84.0</td>
</tr>
<tr>
<td>seq</td>
<td>-10.4</td>
<td>-25.6</td>
<td>-2.94</td>
<td>-0.5</td>
<td>-14.9</td>
<td>90.5</td>
</tr>
<tr>
<td>spla</td>
<td>-7.16</td>
<td>-10.9</td>
<td>-2.91</td>
<td>-0.2</td>
<td>-10.0</td>
<td>75.1</td>
</tr>
<tr>
<td>tseng</td>
<td>-11.2</td>
<td>-23.0</td>
<td>-5.32</td>
<td>-8.3</td>
<td>-17.9</td>
<td>96.8</td>
</tr>
<tr>
<td>Average</td>
<td>-9.05</td>
<td>-15.2</td>
<td>-4.24</td>
<td>-4.2</td>
<td>-13.5</td>
<td>86.7</td>
</tr>
</tbody>
</table>
• Taking the largest benchmark “clma” as an example
• After design tuning, **94.9**% of chips using optimized design can now satisfy the timing constraint. Yield is improved by **89.8**%. 
A combined method of configuration manipulation (rotation and flipping) and LE swap achieves significant timing yield improvement in presence of process variation.
1. Characterization

Process Variation

2. Variation Aware Design

3. Physical Unclonable Function
To develop a unique chip ID using process variations (also known as a PUF)
“Physical One Way Function” (POWF) is published in Science by a MIT group in 2001, which is based on optics.

In 2002, “Physical Unclonbale Function” (PUF) is proposed by another group from MIT. The measurement circuitry and PUF are integrated onto the same electrical circuit and fabricated on silicon.

Applications

- Public-key cryptosystem
- Keyless entry systems and RFID devices
- Digital rights management
Aims

› To develop the circuits on FPGA to obtain a unique chip ID using process variations (also known as a PUF)
› To find a method to counter systematic variation
› To find a method to improve reliability with low cost
Architecture of proposed Chip-ID generator

The hardware cost for proposed chip-ID generator (64bits) is very small.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Consumption</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>65</td>
<td>4,896</td>
<td>1%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>772</td>
<td>4,896</td>
<td>15%</td>
</tr>
<tr>
<td>Number of BUFGMUXs</td>
<td>2</td>
<td>24</td>
<td>8%</td>
</tr>
<tr>
<td>Number of DCMs</td>
<td>1</td>
<td>4</td>
<td>25%</td>
</tr>
</tbody>
</table>
A “Cell” and One-Bit Generation

- Common centroid layout is used to mitigate spatial correlation of process variation.
- Overlapped cell composition improves hardware efficiency.

\[ R_i = (N_A + N_D) - (N_B + N_C) \]

\( R_i \) is the residue value. If \( R_i \) is positive, the bit generated by this cell is 0, otherwise, it is 1.
Residue Distribution with Fixed RO

Source of unstable bits
An configurable RO can be encapsulated into one CLB, with 4-bit configuration, providing 16 different loop paths.
The configuration with maximum absolute value of residue is selected for chip ID generation.
Small residue values are eliminated, therefore the reliability is improved.
Statistical Properties

› Cell configuration:
  - No evident biases towards some particular configurations

› One/zero ratio:
  - 1.06 on average (ideally 1)

› Inter-chip Hamming distance:
  - 47% of the bit width (ideally 50%)
Reliability

› No bit flip is observed under fixed operation condition (1.2v, 20°C)

› Under extreme condition (80°C, 1.2±0.3V), there is only one bit flip over 50000 repetitions.
Conclusion

› Technical novelties
  - Common centroid layout
  - Configurable RO

› Benefits
  - Good statistical properties
  - High reliability over a wide range of operation conditions (voltage & temperature)
  - Low hardware cost
Summary

› Characterization
  - Fine-grained
  - Accurate relative delay estimation

› Variation-aware Design
  - Configuration manipulation effectively counters systematic variation (spatial correlation)
  - LE Swap effectively counters random variation

› Physically Unclonable Function
  - Common centroid layout minimizes systematic variation
  - Configurable RO improves reliability with low cost
Future Work

› Characterization
  - Full characterization for a LUT
  - Accurate region-based variation evaluation for FPGA interconnect circuits

› Variation-aware Design
  - Evaluation on non-symmetric architecture
  - Evaluation on commercial FPGAs

› Physically Unclonable Function
  - Parallelism for fast ID generation
Open Questions

› How to develop FPGA architectures which can be fully self-characterised?
› Can we develop practical post-layout variation-aware techniques to improve timing performance?
› Explore coding and other techniques to produce a guaranteed-reliable chip ID?