Challenges and Opportunities of ESL Design Automation

Zhiru Zhang*, Deming Chen+

*AutoESL Design Technologies, Inc.
+ECE/University of Illinois, Urbana-Champaign
Outline

• Introduction
• Opportunities and Challenges
• Modeling
• Synthesis and Optimization
  – Advanced Memory Synthesis
  – Effective Power Analysis and Optimization
  – Variation-Aware High-Level Synthesis
• Conclusions
Introduction

• The rapid increase of design complexity urges the design community to raise the level of abstraction beyond RTL.

• Electronic system-level (ESL) design automation has been widely identified as the next productivity boost for the semiconductor industry.

• High-level synthesis (HLS) is a key cornerstone of ESL design automation.

• However, the transition to ESL design will not be as well accepted as the transition to RTL in the early 1990s unless
  – robust analysis and synthesis technologies can be built to produce high-quality architectures
  – highly optimized implementations can be automatically generated
Opportunities

• ESL models and tools offer
  – early embedded software development
  – architecture modeling
  – design space exploration
  – rapid prototyping

• HLS fits in nicely for architecture exploration and rapid prototyping
  – early performance/area/power estimations & analyses
  – allows system architects explore different architectures efficiently
  – automated flows to map to an FPGA-based system for system emulation, functional validation and real-time debugging
Challenges - Modeling

- Most efficient virtual platform modeling may not be fully synthesizable
- How to maintain a single synthesizable model as the golden reference for both simulation and synthesis?

- Software centric
  - Optimized for simulation

- HW centric
  - Optimized for implementation
Challenges - Analysis and Optimization (1)

- Efficient support of the memory hierarchy and memory optimization
  - limited memory ports often become the performance bottleneck
  - oversized memory blocks would create wiring detours and routability problem

- Accurate high-level power and performance analysis
  - sophisticated activity propagation
  - clock tree with clock gating
  - multi-voltage islands, dynamic voltage frequency scaling, and power gating
  - low-level physical implementations
  - interconnect
Challenges - Analysis and Optimization (2)

• Effective power and performance optimization
  – large design space
  – most of the problems are NP-hard
  – scheduling, binding and resource allocation are interdependent
  – parallelism extraction
  – quality convergence of layout-driven synthesis

• Process variation
  – variation modeling at high level
  – yield analysis and optimization
Challenges - Others

• HLS for reliability
• HLS for thermal optimization
• ECO
• Verification
• IP integration
• ...

Modeling – Dynamic Behavior and Standardization

• The synthesis tool shall continue to improve to handle a broader class of language constructs.
  – support dynamic behaviors in certain restricted forms.
  – extract the static binding and connectivity from the seemingly dynamic specifications.
  – extend and enhance the predominant static analysis methods.

• The design community and synthesis tool providers shall converge to a standard synthesizable subset.
  – On top of the standard, industry and academia shall collaborate to make available a set of reusable templates and libraries as references for efficient synthesis of common design patterns.
  – The reference templates and libraries should be relatively efficient in execution time and memory footprint.
Modeling - Separation of Functionality and Constraints

- Synthesize hardware details from target-neutral source code
  - Golden functional spec for reuse
  - Technology/platform-dependent RTLs
  - Synthesis influenced by separated constraints & directives
Advanced Memory Synthesis

• On-chip memory partitioning for throughput optimization [Cong, et al., ICCAD’09]
• Support of efficient memory hierarchies including automatic caching and prefetching [Putnam, et al. ISCA’09]
• Communication overlapping with computation
• Efficient access to external memories shared by the host processor and accelerator
A Case Study: Loop Pipelining

- Computation kernels are captured by perfect loop nests
- Loop pipelining allows a new iteration to begin processing before the previous iteration completes
  - Initiation interval (II) : number of time steps before the next iteration begin processing
  - Performance limitation
    - Loop carried dependence
    - Resource constraints

```
for (i = 2; i < N; i++)
```

Pipelining with II=1 is infeasible using a dual-port memory

Courtesy: [Cong, et al., ICCAD’09]
Motivation Example

Scheduling can affect memory partitioning

Generates optimal memory partitioning solutions integrated with scheduling problem

Courtesy: [Cong, et al., ICCAD’09]
Experimental Results (Throughput)

Platform: xilinx Virtex-4 FPGA

<table>
<thead>
<tr>
<th>Function</th>
<th>Original II</th>
<th>AMP II</th>
<th>Original Slices</th>
<th>AMP Slices</th>
<th>COMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>fir</td>
<td>3</td>
<td>1</td>
<td>241</td>
<td>510</td>
<td>2.12</td>
</tr>
<tr>
<td>idct</td>
<td>4</td>
<td>1</td>
<td>354</td>
<td>359</td>
<td>1.01</td>
</tr>
<tr>
<td>litho</td>
<td>16</td>
<td>1</td>
<td>1220</td>
<td>2066</td>
<td>1.69</td>
</tr>
<tr>
<td>matmul</td>
<td>4</td>
<td>1</td>
<td>211</td>
<td>406</td>
<td>1.92</td>
</tr>
<tr>
<td>motionEst</td>
<td>5</td>
<td>1</td>
<td>832</td>
<td>961</td>
<td>1.16</td>
</tr>
<tr>
<td>palindrome</td>
<td>2</td>
<td>1</td>
<td>84</td>
<td>65</td>
<td>0.77</td>
</tr>
<tr>
<td><strong>avg</strong></td>
<td><strong>5.67x</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>1.45</strong></td>
</tr>
</tbody>
</table>

Average 6x performance improvement with 45% area overhead

Courtesy: [Cong, et al., ICCAD'09]
Effective Power Analysis and Optimization

• Three case studies
  – FPGA power estimation and optimization [Chen, et al., ASPDAC’07]
  – Scheduling with Soft Constraints, [Cong, et al., ICCAD’09]
  – Variation-Aware, Layout Driven HLS for Performance Yield Optimization [Lucas, et al., ASPDAC’09]
Case 1: Area Characterization

FPGA power estimation relies on area characterization

<table>
<thead>
<tr>
<th>Operation</th>
<th>Resource</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add/Subtract</td>
<td>LE</td>
<td>$N$</td>
</tr>
<tr>
<td>Bitwise and/or/xor</td>
<td>LE</td>
<td>$N$</td>
</tr>
<tr>
<td>Compare (=, &gt;, ≥)</td>
<td>LE</td>
<td>round($0.67*N + 0.62$)</td>
</tr>
<tr>
<td>Shift (with variable shift distance)</td>
<td>LE</td>
<td>round($0.045<em>N^2 + 3.76</em>N - 8.22$)</td>
</tr>
<tr>
<td>Multiply</td>
<td>DSP9x9</td>
<td>$N \leq 18: \lceil N/9 \rceil$; $N \leq 36: \lceil N/18 \rceil$</td>
</tr>
<tr>
<td>Multiplexer</td>
<td>LE</td>
<td>$N \times \text{round}(0.67*K)$</td>
</tr>
</tbody>
</table>

$N$ and $K$ represent the bitwidth and the number of input operands, respectively.

Target Altera Stratix FPGAs in this work
## Delay Characterization

Delay characterization to study power/delay tradeoff

<table>
<thead>
<tr>
<th>Operation</th>
<th>Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add/Subtract</td>
<td>0.024*N+1.83</td>
</tr>
<tr>
<td>Bitwise and/or/xor</td>
<td>&lt; 2</td>
</tr>
<tr>
<td>Compare (=, &gt;, ≥)</td>
<td>0.014*N+2.14</td>
</tr>
<tr>
<td>Shift (with variable shift distance)</td>
<td>4.3<em>10^{-5}N^5-5</em>10^{-3}N^2+0.24*N+0.93</td>
</tr>
<tr>
<td>Multiply</td>
<td>[N \leq 9: 3.05] [N \leq 18: 3.83] [N \leq 36: 7.69]</td>
</tr>
<tr>
<td>Multiplexer (8-to-1)</td>
<td>9.8<em>10^{-5}N^5-7.4</em>10^{-3}N^2+0.2*N+1.07</td>
</tr>
</tbody>
</table>
Design Space Exploration

Node 2: (1) (2) two mul
   (1, 2) one mul

Node 3: (1) (2) (3) three mul
   (1, 2) (3) two mul
   (1, 3) (2) two mul

Node 4: (1) (2) (3) (4)
   (1, 2, 4) (3)
   (1, 2) (3, 4)
   (1, 2) (3) (4)
   (1, 3, 4) (2)
   (1, 3) (2, 4)
   (1, 3) (2) (4)

Datapath for solution (1, 2, 4) (3)

MUXes

registers

pruned
Power and Performance Comparison

![Power & Fmax Comparison](image)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ratio</th>
<th>Power</th>
<th>Fmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImXRLF</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ImXRLF-Power</td>
<td>0.97</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>xPlore-Power</td>
<td>0.68</td>
<td></td>
<td>1.16</td>
</tr>
</tbody>
</table>
Case 2: Operation Gating

```cpp
int module(int A, int B) {
    int B2 = B * B;
    bool c = A > 0;
    int r = c ? A : B2;
    return r;
}
```

- Schedule to maximize the gating/shutdown opportunities.
- Use constraints to enforce node orders?
Slack Optimization

- Slack within a clock cycle is desirable.
- Add a constraint to separate nodes when slack is too small?
  - What if latency constraint is very tight?
Our approach provides

• **33.9% power reduction** compared to baseline on average
• **17.1% power reduction** compared to Chen’s method on average
• Close result to the ILP method
Case 3: Process Variation and Its Effect

- Process variation increases as device and interconnect feature sizes are scaled down
  - 30% performance variation and 5X leakage variation
- Traditional guard-bandung uses pessimistic worst-case process corners
  - Inefficient as the variability increases with scaling

(Source: Intel)

(Source: IBM)
FastYield Algorithm Overview

Scheduled CDFG

Initial Register Allocation and Binding

SSTA Driven Floorplanner

Improvement? Yes → Rebinding

Improvement? No → Bound Benchmark
Timing Driven Floorplanner

- Modified version of the simulated annealing based Parquet floorplanner
- A statistical timing analysis is performed after 5 SA moves
  - Minimize the sum of the mean and standard deviation
- Cost function:

\[
Z \sim N(\mu_z, \sigma_z) = \max(\text{reg}_1(\mu_1, \sigma_1), \text{reg}_2(\mu_2, \sigma_2), \ldots, \text{reg}_n(\mu_n, \sigma_n))
\]

\[
T_R = \frac{\mu_z + \sigma_z}{\mu_{\text{best}} + \sigma_{\text{best}}}
\]

\[
\text{Cost} = \alpha \cdot \text{area} + \beta \cdot T_R
\]

- PCA based SSTA
  - Interconnects modeled as 2 pin nets with Manhattan distance length.
  - Unit correlation model
Unit Correlation Model

- Correlation is based on the distance between the unit centerpoints.
- Matches high level unit characterization.
- Correlation matrix used in PCA SSTA with $\sigma_{\text{inter}}$.

Sample floorplan
One benchmark - chem

- Improvement of FastYield comes from two factors:
  - the mean of the pdf has been shifted to a lower clock value.
  - the variance has been reduced.
- A significant PY jump for a relatively minor change in the mean clock period
## FastYield Results

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>BindBWM 85% Yield Clk (ns)</th>
<th>PY at FY Rebind 85% Yield Clk (%)</th>
<th>FastYield Initial 85% Yield Clk (ns)</th>
<th>PY at FY Rebind 85% Yield Clk (%)</th>
<th>Total FY Run Time (min)</th>
<th>FY Rebind reduction in Clk over BindBWM (%)</th>
<th>FY Rebind 85% PY Gain over BindBWM (%)</th>
<th>FY Rebind reduction in Clk over FY Initial (%)</th>
<th>FY Rebind 85% PY Gain over FY Initial (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>chem</td>
<td>6.9</td>
<td>12.5</td>
<td>6.1</td>
<td>67.7</td>
<td>6.0</td>
<td>75</td>
<td>14.17</td>
<td>72.5</td>
<td>2.35</td>
</tr>
<tr>
<td>dir</td>
<td>5.8</td>
<td>1.5</td>
<td>4.9</td>
<td>70.9</td>
<td>4.8</td>
<td>43</td>
<td>16.71</td>
<td>83.5</td>
<td>1.76</td>
</tr>
<tr>
<td>honda</td>
<td>5.7</td>
<td>8.1</td>
<td>4.9</td>
<td>82.6</td>
<td>4.9</td>
<td>28</td>
<td>14.39</td>
<td>76.9</td>
<td>0.32</td>
</tr>
<tr>
<td>mcm</td>
<td>4.9</td>
<td>11.4</td>
<td>4.3</td>
<td>78.0</td>
<td>4.2</td>
<td>40</td>
<td>14.57</td>
<td>73.6</td>
<td>3.34</td>
</tr>
<tr>
<td>pr</td>
<td>5.2</td>
<td>0.1</td>
<td>4.5</td>
<td>70.1</td>
<td>4.3</td>
<td>24</td>
<td>16.47</td>
<td>84.9</td>
<td>3.04</td>
</tr>
<tr>
<td>steam</td>
<td>6.2</td>
<td>7.6</td>
<td>5.5</td>
<td>76.3</td>
<td>5.5</td>
<td>64</td>
<td>11.88</td>
<td>77.4</td>
<td>1.14</td>
</tr>
<tr>
<td>wang</td>
<td>5.3</td>
<td>1.6</td>
<td>4.7</td>
<td>80.8</td>
<td>4.6</td>
<td>16</td>
<td>13.29</td>
<td>83.4</td>
<td>0.95</td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14.50</td>
<td>78.9</td>
<td>1.84</td>
</tr>
</tbody>
</table>
Conclusions

- This paper identified a set of critical needs and key challenges in ESL design automation with special focus on HLS
  - software-centric ESL modeling
  - optimizations of memory hierarchy and access
  - power and performance analysis and optimization
  - process variation-aware HLS
- These needs and challenges have created many new and important research directions as well as business opportunities in the EDA community
Acknowledgement

• Students at UIUC and UCLA
• Researchers at AutoESL

• Various funding agencies
  – NSF, SRC, GSRC, Altera, Intel, Magma, Xilinx
Thank You