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

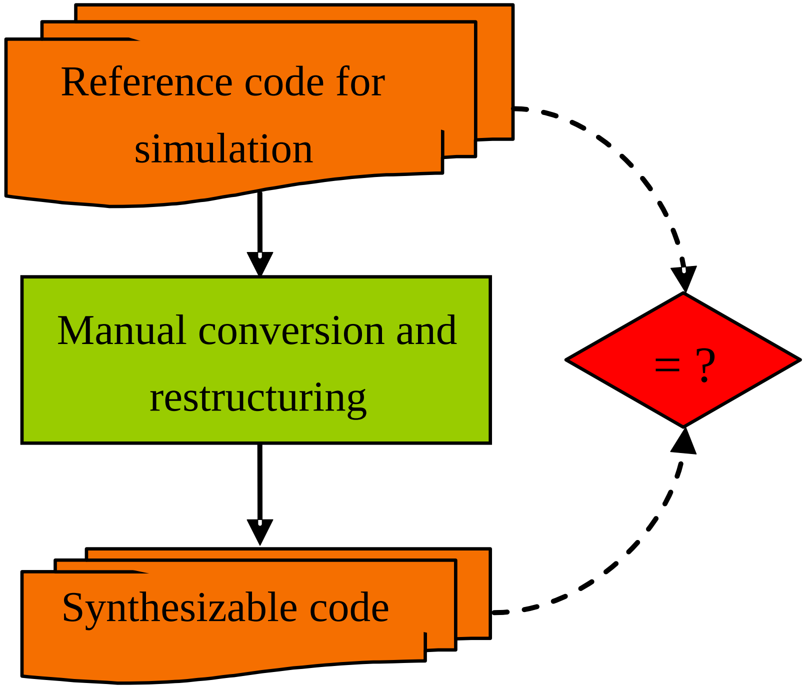
Opportunities

- 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

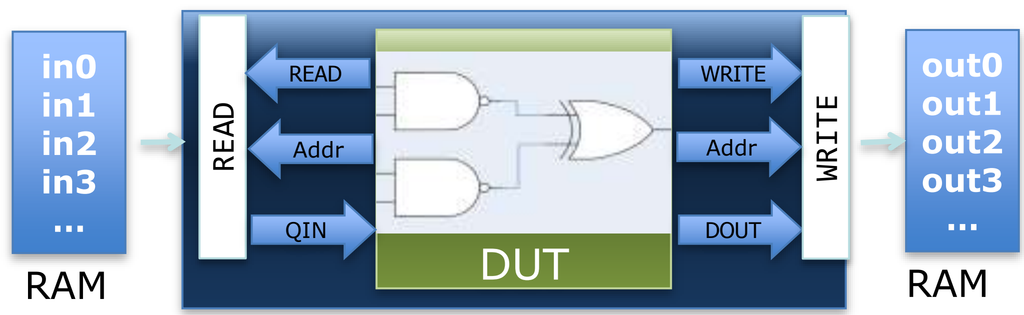
- Synthesize hardware details from target-neutral source code
  - Golden functional spec for reuse
  - Technology/platform-dependent RTLs
  - Synthesis influenced by separated constraints & directives

Source code (What)

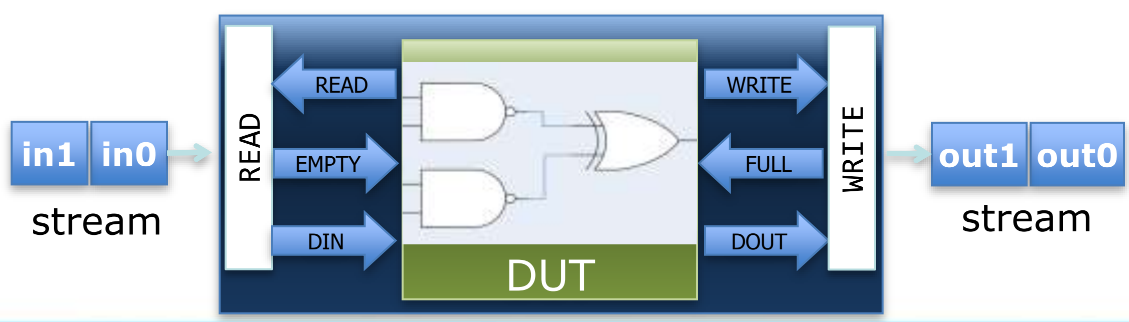
```
void DUT(int in[N], int out[N])
{ … }
```

Constraint/directive (How)

```
set_interface –type memory –port {in out}
```




```
set_interface –type stream –port {in out}
```



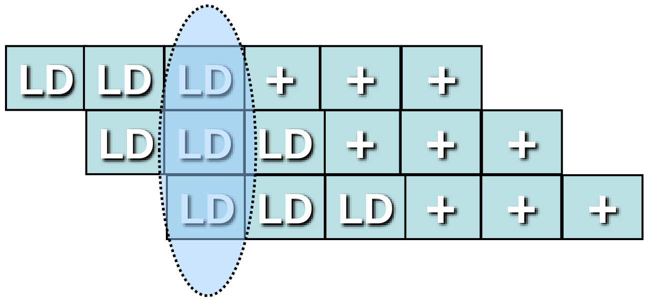
Advanced Memory Synthesis

- 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

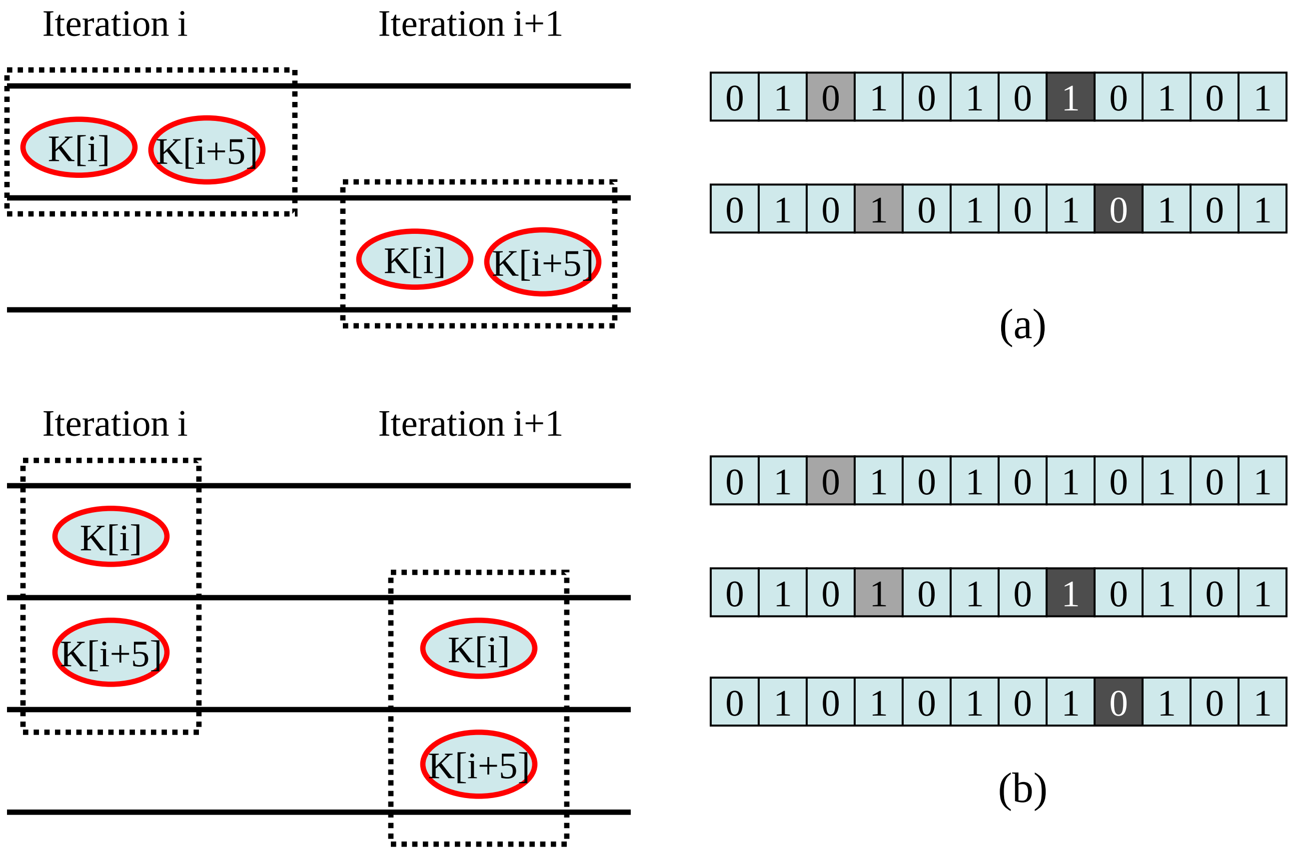
- Computation kernels are captured by perfect loop nests
- Loop pipelining allows a new iteration to begin processing before the previous iteration completes
  - Initiation interval (II) : number of time steps before the next iteration begin processing
  - Performance limitation
    - Loop carried dependence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

```
for (i = 2; i < N; i++)
   sum += A[i] + A[i-1] + A[i-2];
```



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

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

	Original II	AMP II	Original Slices	AMP Slices	COMP
fir	3	1	241	510	2.12
idct	4	1	354	359	1.01
litho	16	1	1220	2066	1.69
matmul	4	1	211	406	1.92
motionEst	5	1	832	961	1.16
palindrome	2	1	84	65	0.77
avg		5.67x			1.45

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

Operation	Resource	Usage
Add/Subtract	LE	N
Bitwise and/or/xor	LE	N
Compare (=, >, ≥)	LE	$round(0.67*N+0.62)$
Shift (with variable shift distance)	LE	$round(0.045*N^2+3.76*N-8.22)$
Multiply	DSP9x9	$N \leq 18: \lceil N/9 \rceil$ $N \leq 36: \lceil N/18 \rceil$
Multiplexer	LE	$N*round(0.67*K)$

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

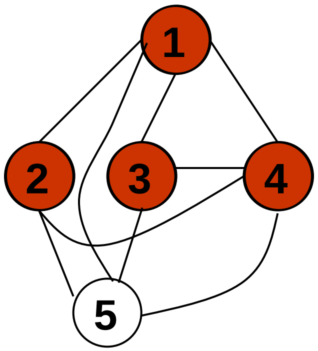
Target Altera Stratix FPGAs in this work

Delay Characterization

Delay characterization to study power/delay tradeoff

Operation	Delay (*ns*)
Add/Subtract	$0.024*N+1.83$
Bitwise and/or/xor	< 2
Compare ($=, >, \geq$)	$0.014*N+2.14$
Shift (with variable shift distance)	$4.3*10^{-5}*N^3-5*10^{-3}*N^2+0.24*N+0.93$
Multiply	$N \leq 9$: 3.05 $N \leq 18$: 3.83 $N \leq 36$: 7.69
Multiplexer (8-to-1)	$9.8*10^{-5}*N^3-7.4*10^{-3}*N^2+0.2*N+1.07$

Design Space Exploration



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

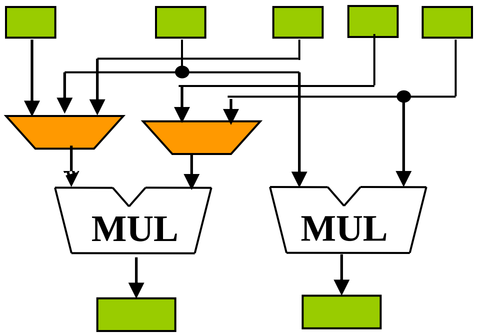
Node 3: (1) (2) (3) three mul
(1, 2) (3) two mul
(1, 3) (2) two mul

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)

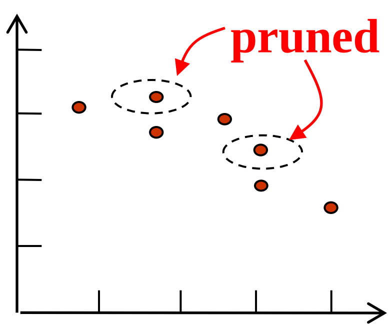
....

registers

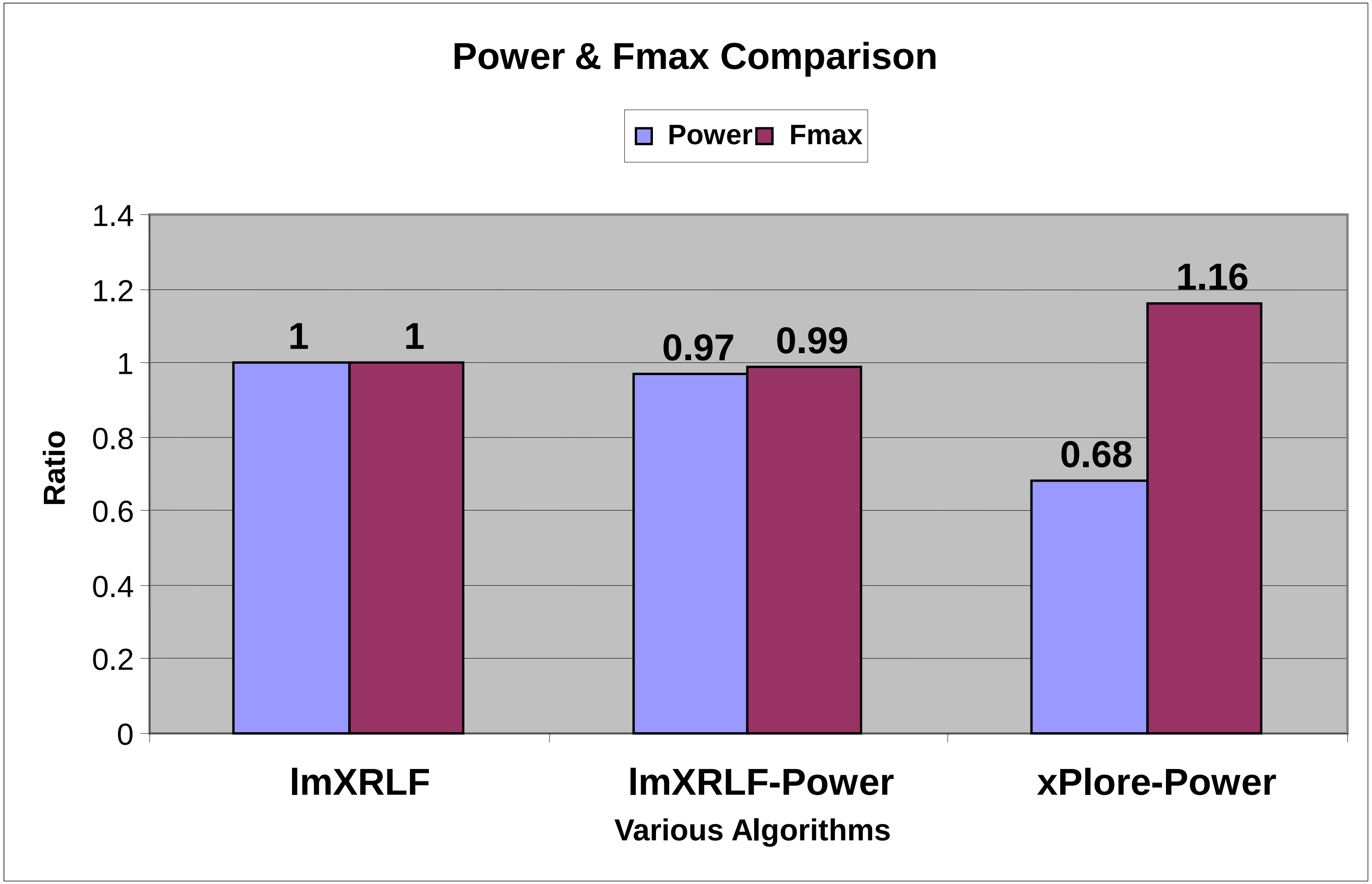
MUXes



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

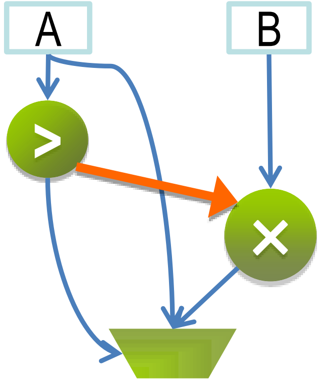


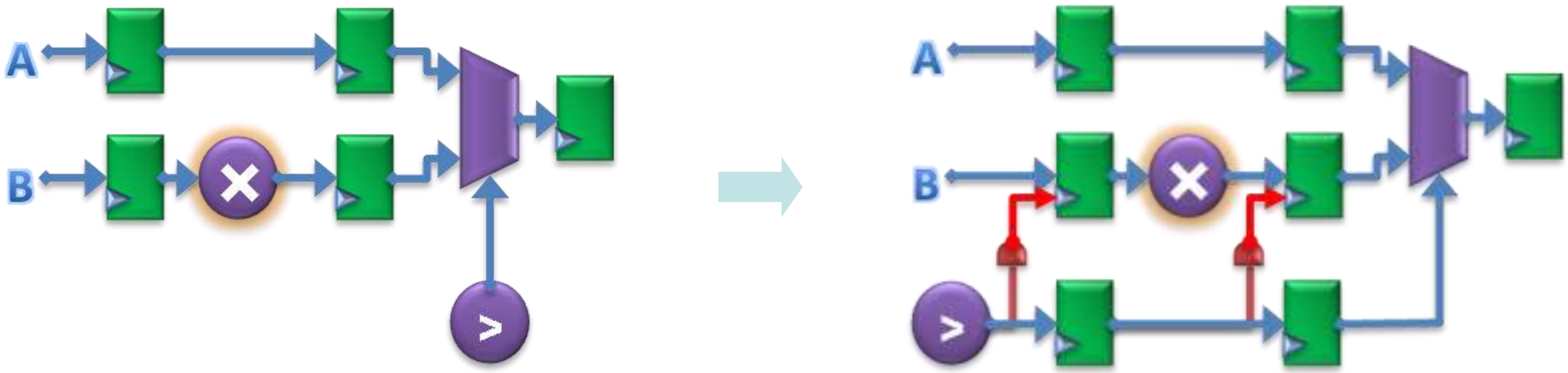
Power and Performance Comparison



Case 2: Operation Gating

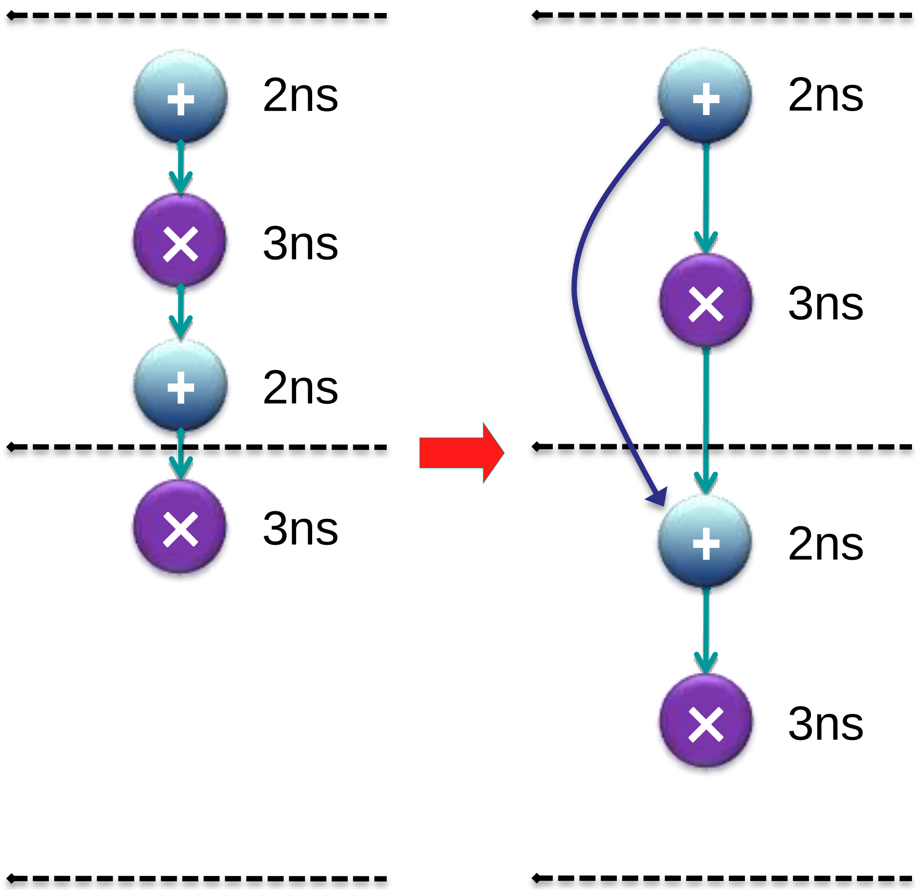
```
int module(int A, int B)
{
    int B2 = B * B;
    bool c = A > 0;
    int r = c ? A : B2;
    return r;
}
```





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

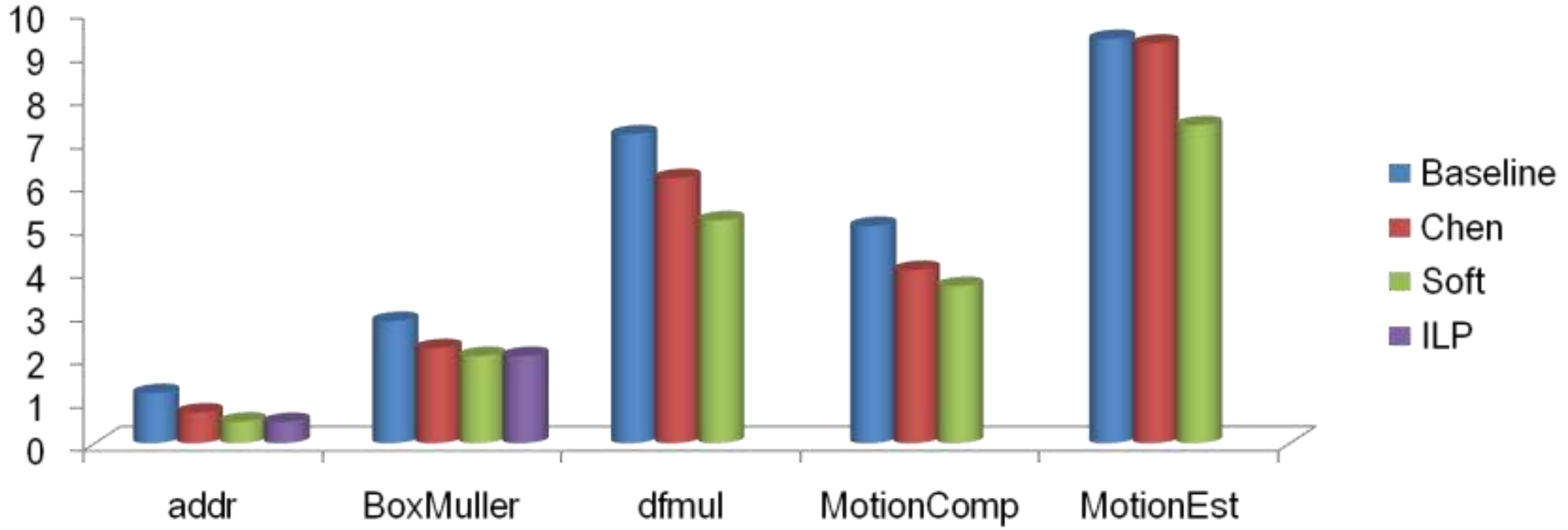
Slack Optimization



Cycle time	7ns
Latency budget	2 cycles

- 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?

Comparison on Power

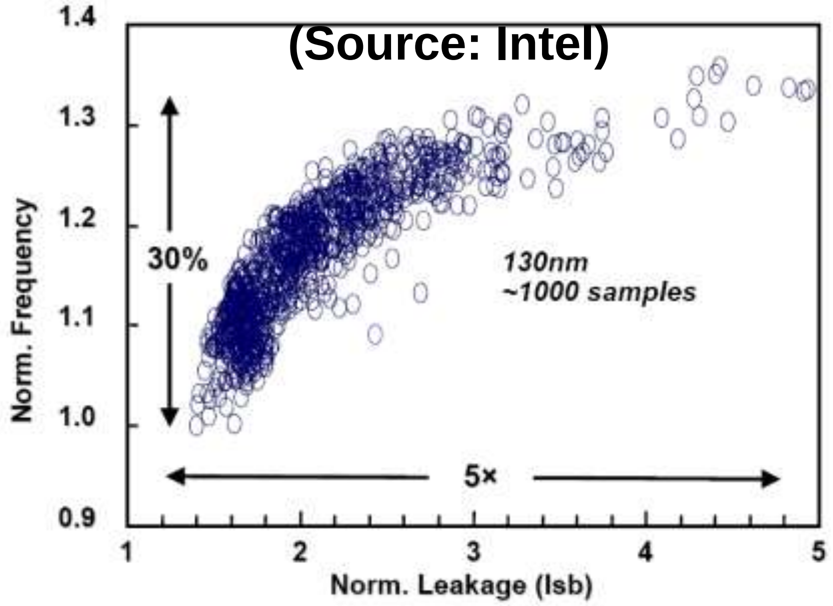


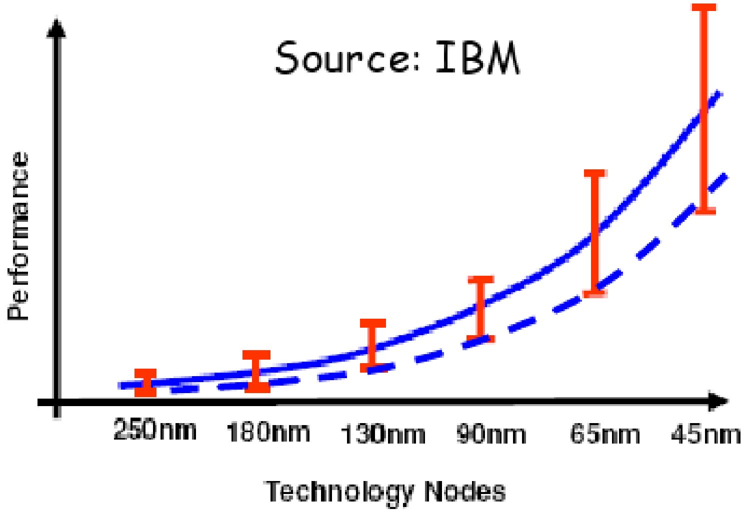
Our approach provides

- **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

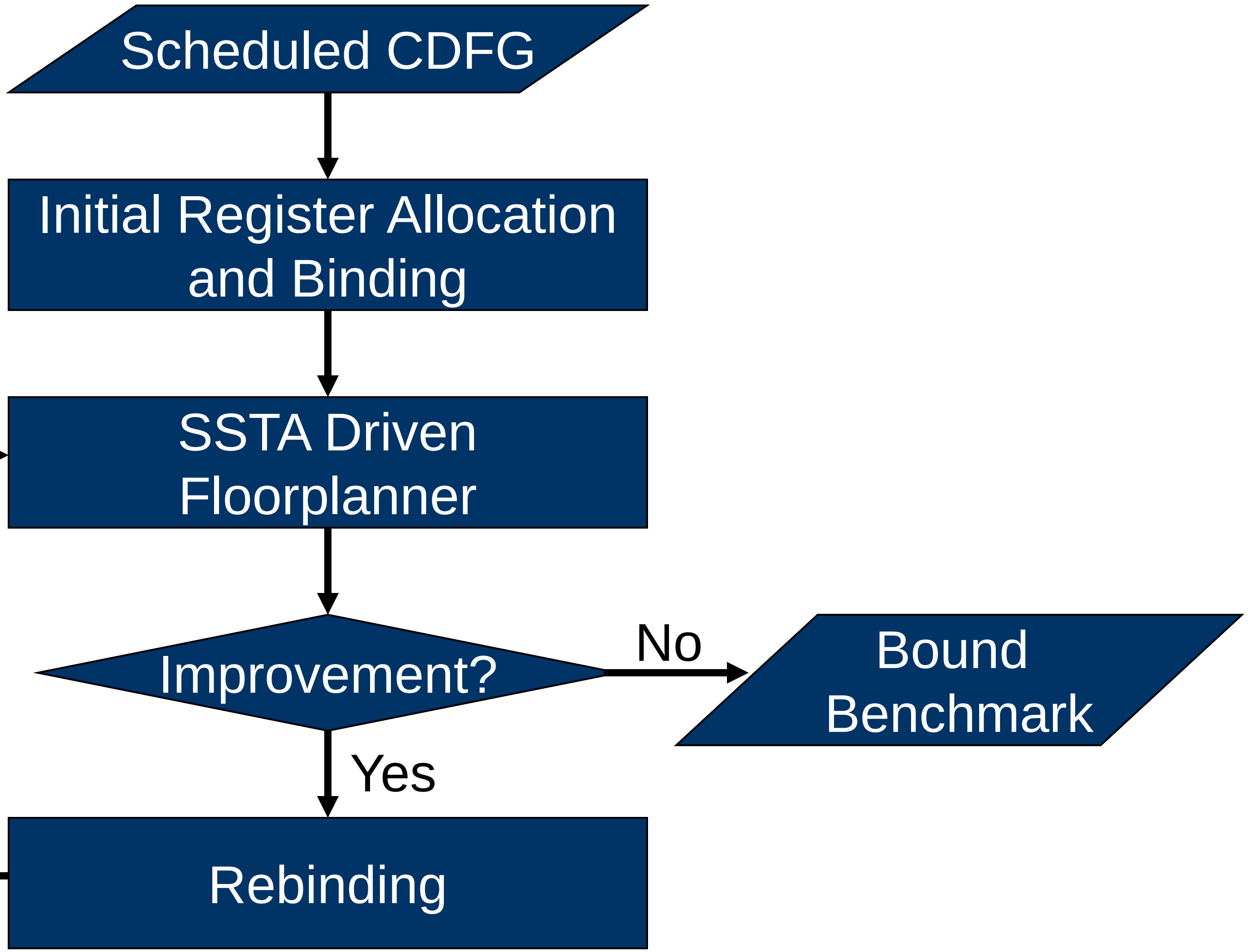
Case 3: Process Variation and Its Effect

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





FastYield Algorithm Overview



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:

$$Z \sim N(\mu_z, \sigma_z) = \max(reg_1(\mu_1, \sigma_1), reg_2(\mu_2, \sigma_2), \ldots, reg_n(\mu_n, \sigma_n))$$

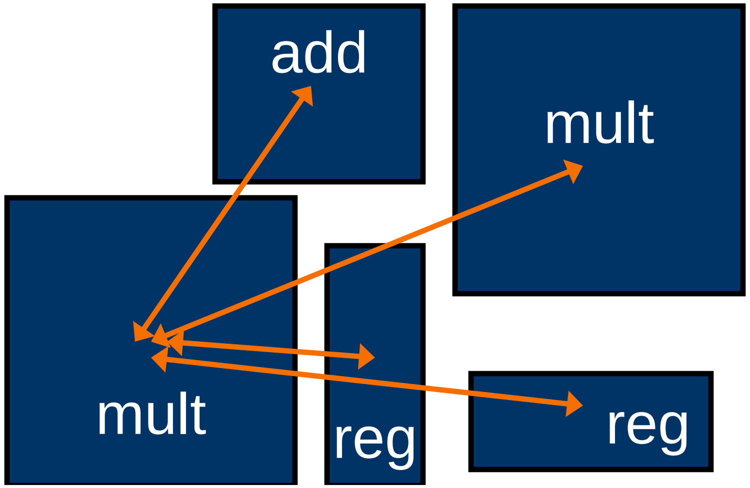
$$T_R = \frac{\mu_z + \sigma_z}{\mu_{best} + \sigma_{best}}$$

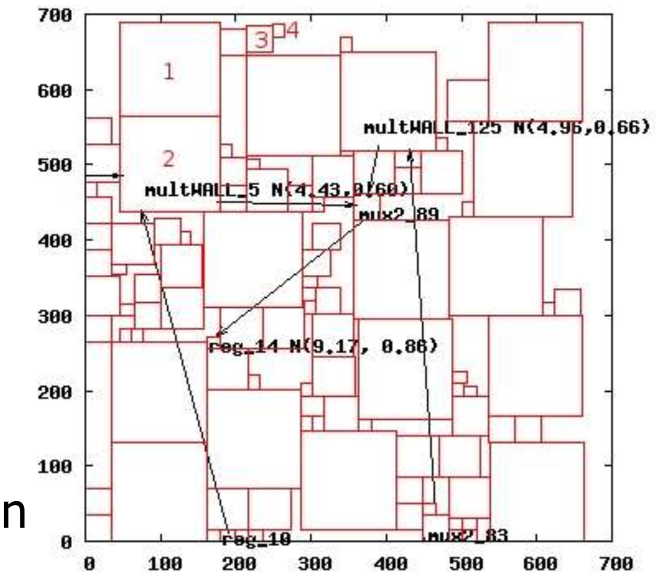
$$Cost = \alpha * area + \beta * T_R$$

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

Unit Correlation Model

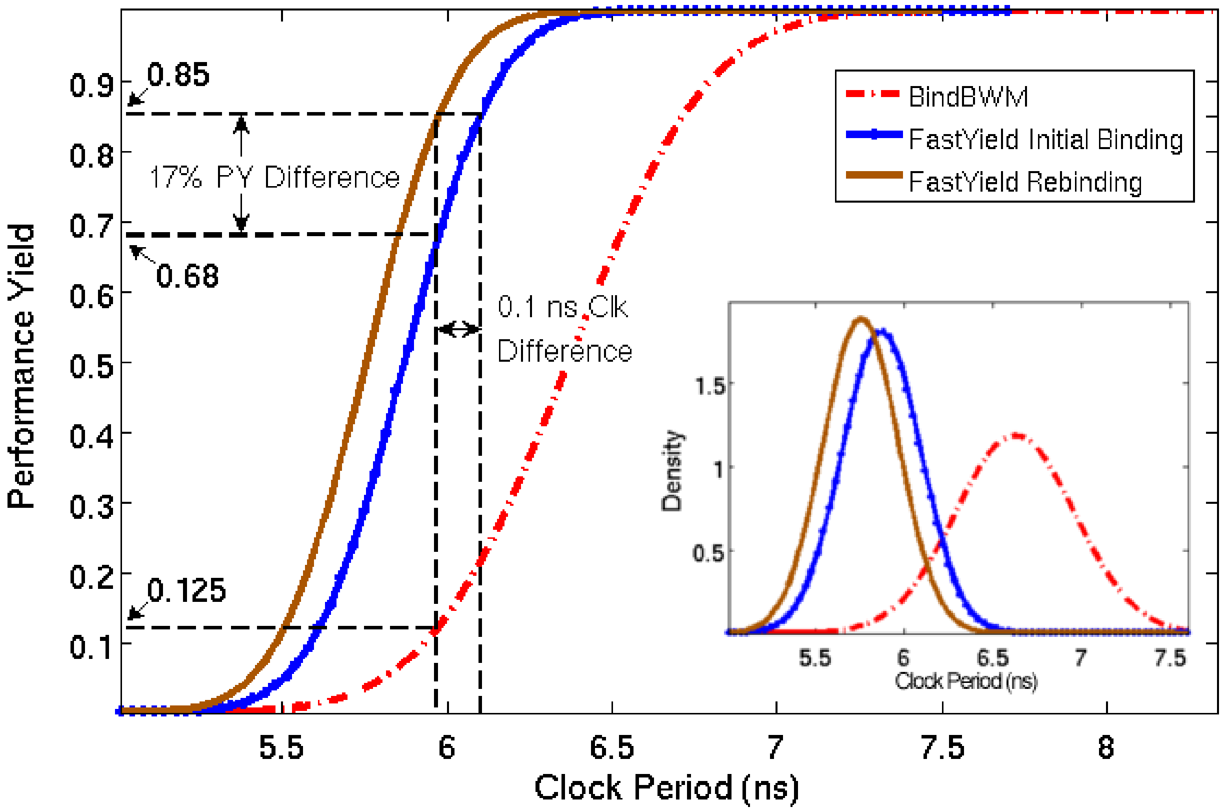
- Correlation is based on the distance between the unit centerpoints
- Matches high level unit characterization
- Correlation matrix used in PCA SSTA with σ_{inter}





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

	BindBWM		FastYield Initial		FastYield Rebind		Comparison			
Bench mark	85% Yield Clk (ns)	PY at FY Rebind 85% Clk (%)	85% Yield Clk (ns)	PY at FY Rebind 85% Clk (%)	85% Yield Clk (ns)	Total FY Run Time (min)	FY Rebind reduction in Clk over BindBWM (%)	FY Rebind 85% PY Gain over BindBWM (%)	FY Rebind reduction in Clk over FY Initial (%)	FY Rebind 85% PY Gain over FY Initial (%)
chem	6.9	12.5	6.1	67.7	6.0	75	14.17	72.5	2.35	17.3
dir	5.8	1.5	4.9	70.9	4.8	43	16.71	83.5	1.76	14.1
honda	5.7	8.1	4.9	82.6	4.9	28	14.39	76.9	0.32	2.4
mcm	4.9	11.4	4.3	78.0	4.2	40	14.57	73.6	3.34	7.0
pr	5.2	0.1	4.5	70.1	4.3	24	16.47	84.9	3.04	14.9
steam	6.2	7.6	5.5	76.3	5.5	64	11.88	77.4	1.14	8.7
wang	5.3	1.6	4.7	80.8	4.6	16	13.29	83.4	0.95	4.2
Avg.							14.50	78.9	1.84	9.8

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