Micro-architecture Pipelining Optimization with Throughput-Aware Floorplanning

Yuchun Ma* Zhuoyuan Li* Jason Cong** Xianlong Hong* Glenn Reinman** Sheqin Dong* Qiang Zhou*

*Department of Computer Science & Technology, Tsinghua University, Beijng 100084, P.R.China
**Department of Computer Science, UCLA, USA

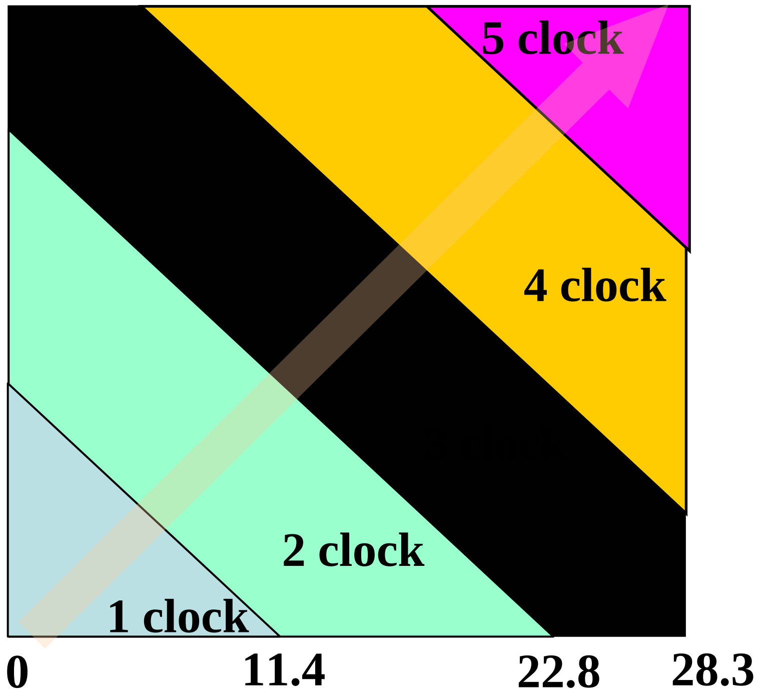
Outline

Outline

- Introduction
- Problem Formulation
- Simultaneous Block and Interconnect Pipelining
  - MILP formulation
  - Graph-based heuristic algorithm
- Throughput aware floorplanning with pipelining
- Case Study for a Design Driver
- Conclusions and Future Works

Effects of Wire Delay in Pipelined Design

- In DSM regime, Single-cycle full chip communication will be no longer possible.
- Deeper superscalar pipelines increased latencies and a significant degradation in instruction throughput



- ITRS'01 0.07um Tech
- 5.63GHz across-chip clock
- 800mm^2 (28.3mm x 28.3mm)
- IPEM BIWS estimations
  - Buffer size: 100x
  - Driver/receiver size: 100x
- Pipelining
  - Need 5 clock cycles from corner to corner
  - Performance reduced by a factor of up to 2 to 3

Throughput-aware Strategies

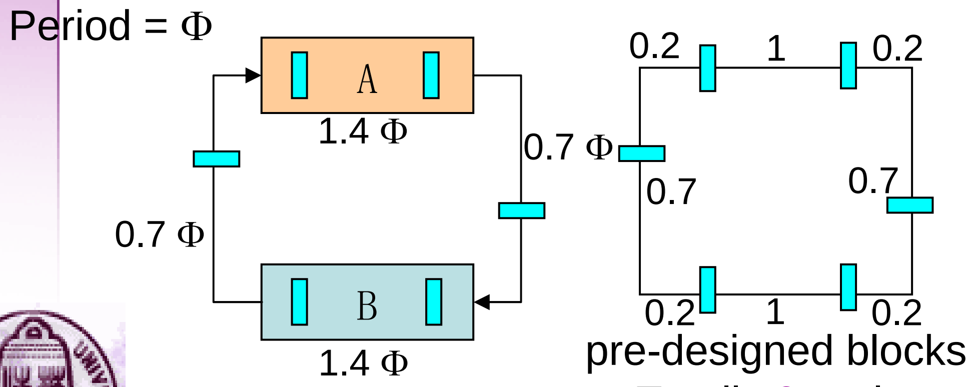
- Physical design should focus on keeping the throughput-critical paths as short as possible;
- Provide feedback to architects at very early design stages.
  - Throughput-aware design at the floorplanning level
  - Architecture pipelining with physical design

Previous Works

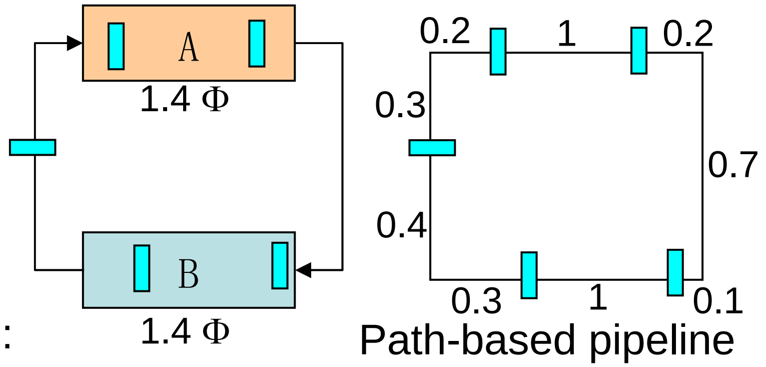
- Throughput-aware floorplanning
  - [Cong, DAC03 and ASPDAC05]: the first automated microarchitecture exploration system combined with physical planning;
  - [Long, DAC04]: a throughput lookup table (LUT);
  - [Ekpanyapong, DAC04]: minimize a weighted sum of bus latencies;
  - [Nookala, DAC05]: a statistical design of experiments which identify the performance critical buses in a micro-architecture.
- Retiming techniques
  - Move the flip-flops within a circuit while keeping its functionality;
  - Minimize the clock period;
  - The pipeline designs inside blocks are still assumed fixed ;
  - Different from micro-architecture pipelining which minimizes the latencies on critical paths

Pipeline models in previous works

- Wire-pipelining under the assumption that the blocks are separately designed.
  - pipelining designs are often sub-optimal.
- Assume all the paths can be optimized and pipelined independently.
  - Block sharing in different loops.
- It is important to use accurate models which can consider both block and wire pipelining simultaneously.



pre-designed blocks:
Totally 6 cycles



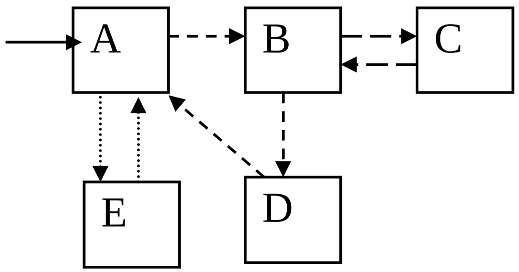
Path-based pipeline
Totally 5 cycles!

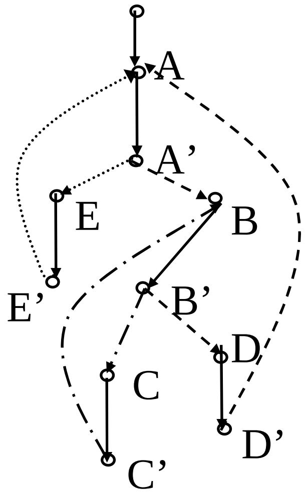
Tsinghua University

Outline

Simultaneous Block and Interconnect Pipelining

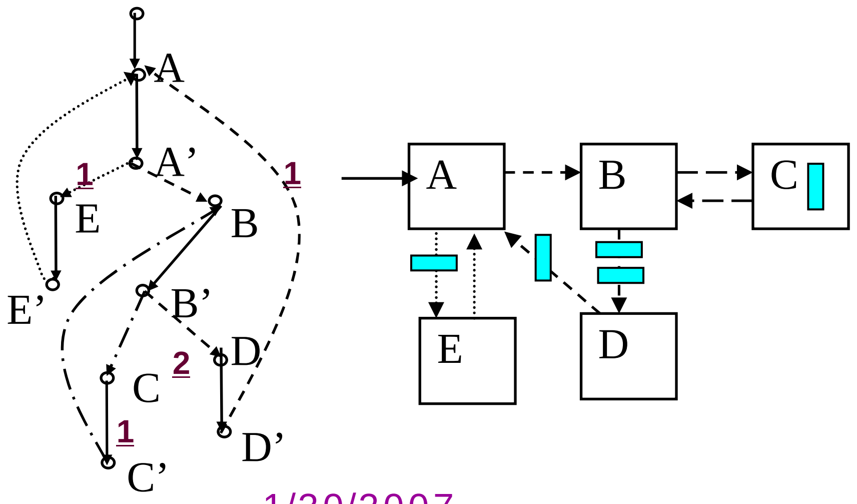
- **Path-based pipelining design should optimize the distribution of flip-flops along multiple paths.**
  - **Represent the micro-architecture design by a *path graph G*(*V*,*E*);**
    - **Treat blocks and wires equally as the components with corresponding delays;**
    - **Each node *v* is the joint between a block and a wire.;**
    - **Each directed edge e_i represents a wire or a block.**





Simultaneous Block and Interconnect Pipelining

- A problem of labeling the edge with the number of flip-flops on each edge.
- Therefore the objective is to find a feasible solution with the optimal performance.
  - Min: the weighted sum of latencies along the critical paths;
  - Constraints: the number of flip-flops should make clock period feasible along any segment of the paths.

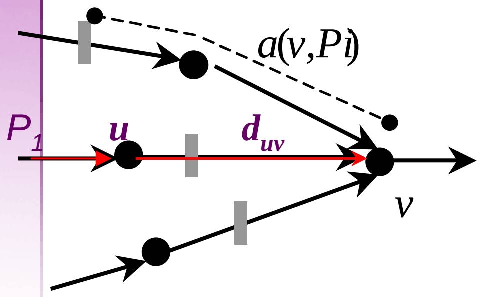


Outline

- Introduction
- Problem Formulation
- Simultaneous Block and Interconnect Pipelining
  - MILP formulation
  - Graph-based heuristic algorithm
- Throughput aware floorplanning with pipelining
- Case Study for a Design Driver
- Conclusions and Future Works

MILP Formulation

- We define a term $a(v, P_i)$ that represents the maximum arrival time at node v along path P_i;
- With the given clock period Φ, the set of paths P with the weight w_{pi} for each path P_i, we can then formulate the problem.



Obj. $\quad Min \sum_{P_i \in P} (w_{Pi} \times \sum_{e_i \in P_i} f_{ei})$

s.t.

$$a(v,P_i) \le \Phi \quad \forall\ v \in V,\ P_i \in P \qquad (1)$$

$$a(v,P_i) \ge 0 \quad \forall\ v \in V,\ P_i \in P \qquad (2)$$

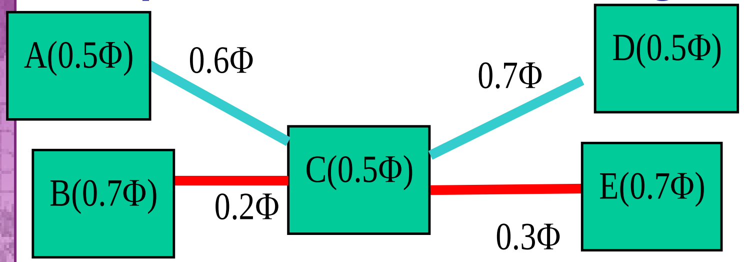
$$f_{ei} \ge 0 \quad \forall\ e_i \in E \qquad (3)$$

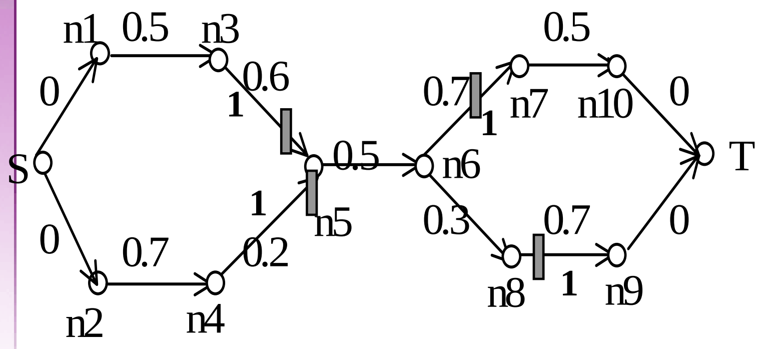
$a(v,P_i) \ge a(u,P_i) + d_{ei} - \Phi * f_{ei} \quad \forall\ e_i \in E$ and e_i is a connection from node u to node v *along* P_i. (4)

$$a(v,P_1) - a(u,P_1) \ge d_{uv} - \phi \times f_{uv}$$

MILP Formulation

P1: A-C-D and P2: B-C-E, each path has the same weight





$Min\ \ f_{13}+f_{35}+f_{56}+f_{67}+f_{710}+f_{24}+f_{45}+f_{56}+f_{68}+f_{89}$

$0 \leq a(1,P1) \leq 1$

$0 \leq a(3,P1) \leq 1$

$0 \leq a(5,P1) \leq 1$

…….

$f_{13} \geq 0$

$f_{35} \geq 0$

…….

$a(3,P1)-a(1,P1) \geq 0.5-f_{13}$

$a(5,P1)-a(3,P1) \geq 0.6-f_{35}$

……

With the values of $a(v,P_i)$ and f_{ei} for all $e_i \in E$, we can place flip-flops accordingly.

Analysis of MILP Approach

- $\sum|P_i|$ real variables $a(v,P_i)$, $|E|$ integer variables f_{ei}, and $2\sum|P_i| + 2|V|$ constraints
- Pros
  - Optimal pipeline
- Cons
  - Time consuming
  - Not applicable to be embedded in floorplanning iterations.

$$\text{Obj.} \quad Min \sum_{P_i\in P}\left(w_{Pi}\times\sum_{e_i\in P_i} f_{ei}\right)$$

$$\text{s.t.} \quad a(v,P_i)\le \Phi \quad \forall\ v\in V,\ P_i\in P \qquad (1)$$

$$a(v,P_i)\ \ge\ 0 \quad \forall\ \ v\in V,\ \ P_i\in P \qquad (2)$$

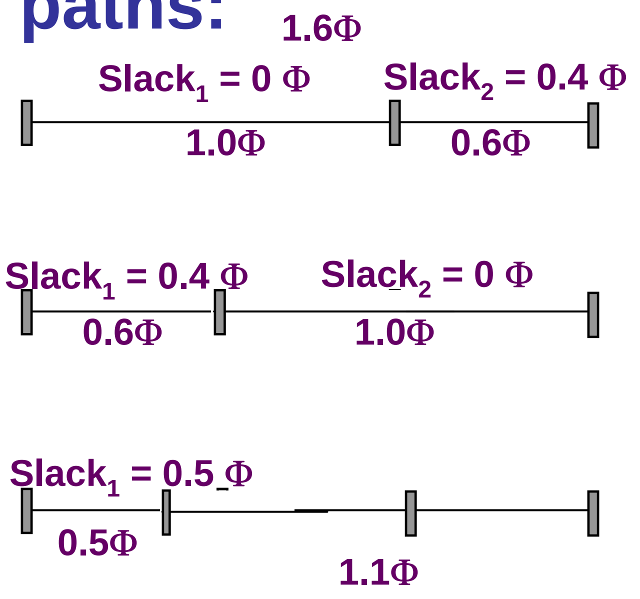
$$f_{ei}\ge 0 \quad \forall\ \ e_i\in E \qquad (3)$$

$$a(v,P_i)\ \ge\ a(u,P_i)\ +\ d_{ei}\ -\Phi * f_{ei}\ \ \forall\ \ e_i\ \in E\ \text{ and } e_i \text{ is a connection from node } u \text{ to node } v \text{ along } P_i. \qquad (4)$$

Slacks along paths

- Slacks along paths:

Slack(P) = 0.4 Φ


1.6Φ
$\text{Slack}_1 = 0\ \Phi$
$\text{Slack}_2 = 0.4\ \Phi$
1.0Φ
0.6Φ
$\text{Slack}_1 = 0.4\ \Phi$
$\text{Slack}_2 = 0\ \Phi$
0.6Φ
1.0Φ
$\text{Slack}_1 = 0.5\ \Phi$
0.5Φ
1.1Φ

$\text{Slack}_1 > \text{Slack(P)}$
One extra cycle!!

- We can optimize the pipeline design by controlling the extra delay slacks when inserting flip-flop along paths.

Dynamic scanning heuristic for combinational circuits

- For a combinational circuit, there is no loop.
  - A directed acyclic graph (DAG) G’(V’,E’);
  - A pair of source node *s* and target node *t* .
- Scan the graph in topology order and try to insert flip-flop to meet the clock period with the least extra cycles.
  - $a(v, P_i)$ is the arrival time for each passing node along each path;
  - During the traversing, the paths are pipelined partially;
    - Delay slacks on the partially pipelined paths;
    - A newly inserted flip-flop will change the timing distribution in the graph;
    - If the delay between a newly inserted flip-flop and the previous flip-flop (or source node) is less than one clock period, it will introduce extra slack.

Tsinghua University

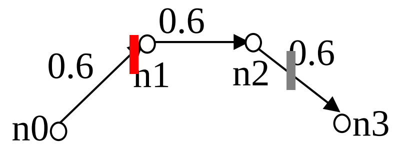
Slack Information

- Ideal Slack along path:
  - if the edges along path are pipelined with no extra slack.

$Ideal\_Slack_{cur}(P) = (2 - 1.8)\Phi = 0.2\Phi$

- Add a new flip-flop *f*: To meet the clock period, each path P_i passing e_i should satisfie:

$$a(u, P_i) + d_{uf} \leq \Phi$$



- The extra slack caused by *f* is

$$Extra\_Slack(P_i, f) = \Phi - (a(u, P_i) + d_{uf})$$

$a(n0,P) = 0$

$d_{0f} = 0.6\Phi$

$Extra\_Slack(P, f) = \Phi - (a(n0,P) + d_{0f}) = 0.4\Phi > Ideal\_Slack_{cur}(P)$

- If $Extra\_Slack(P_i, f) > Ideal\_Slack(P_i)$
  - *f* will generate an extra cycle on path P_i

$Ideal\_Slack(P_i) = \Phi + Ideal\_Slack_{cur}(P_i) - Extra\_Slack(P_i, f)$

one extra cycle

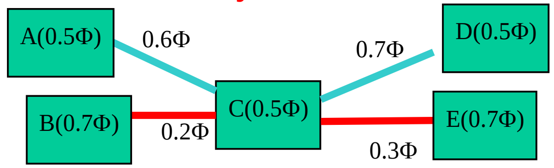
- If $Extra\_Slack(P_i, f) \leq Ideal\_Slack(P_i)$
  - *f* will not generate an extra cycle on path P_i.

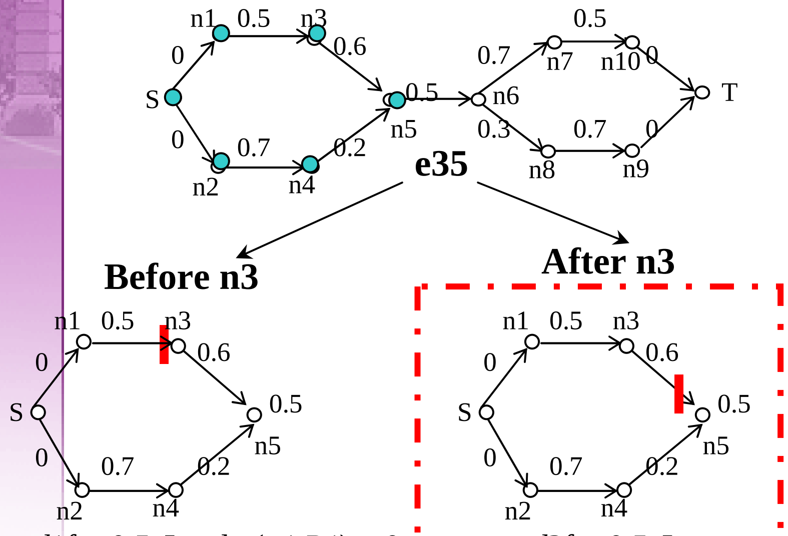
$Ideal\_Slack(P_i) = Ideal\_Slack_{cur}(P_i) - Extra\_Slack(f)$

Dynamic Scanning

Total delay on P1 is 2.8 *Φ*

Total delay on P2 is 2.4 *Φ*



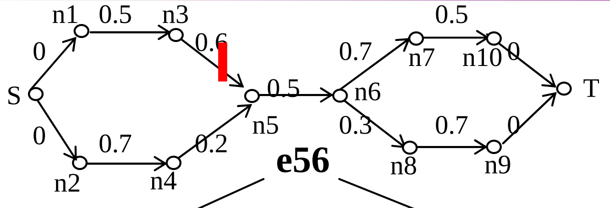


$d1f = 0.5\Phi$ and $a(n1,P1) = 0$
Extra_Slack(*P1*, *f*) = 0.5*Φ* >
Ideal_Slackcur(*P1*)
***one extra cycle* !!**

d3f = 0.5*Φ*;
a(*n3*,*P1*) = 0.5 *Φ*
Extra_Slack(*P1*, *f*) = 0 <
Ideal_Slackcur(*P1*)
***NO extra cycle* !!**

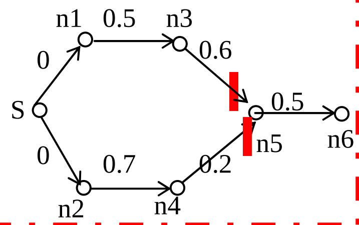
v	a (v, P1)	Ideal Slack (P1)	a (v, P2)	Ideal Slack (p2)	Extra slack	f	duf	Extra cycle
S	**0**	**0.2**	**0**	**0.6**	**-**	×	×	
n1	× **0**	**-**	×	×	**-**	×		×
n2	× **0**	×	**-**		**-**	**-**	**-**	**-**
n3	**0.5**		×	×	**-**	**-**	**-**	**-**
n4	**0.7**	×	**-**		**-**	**-**	**-**	**-**
n5	**1.1** **0.1**	**0.2** **0.2**	**-** **0.9**	**-** **0.6**	**0**	e_{35}	**0.6**	**0**
n6								
n7								
n8								
n9								
n10								
T	-	-	-	-	-	-	-	-

Dynamic Scanning

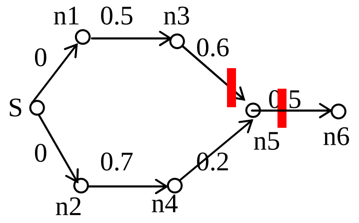


e56

Before n5



After n5



$d4f = 0.2\Phi$ and $a(n4,P2) = 0.9\Phi$

Extra_Slack(P2, f) $= 0.1\Phi <$ *Ideal_Slackcur(P2)*

NO extra cycle !!

$d5f = 0.1\Phi$;

$a(n5,P1) = 0.1\ \Phi$ $a(n5,P2)=0.9\ \Phi$

Extra_Slack(P1, f) $= 0.8\Phi >$ *Ideal_Slackcur(P1)*

One extra cycle on P1 !!

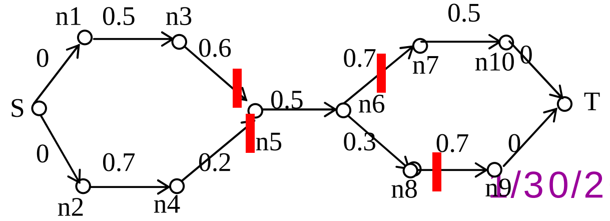
Extra_Slack(P2, f) $= 0\Phi <$ *Ideal_Slackcur(P2)*

NO extra cycle on P2 !!

e67

Before n6

After n6



v	a (v, P1)	Ideal Slack (P1)	a (v, P2)	Ideal Slack (p2)	Extra slack	f	duf	Extra cycle
S	**0**	**0.2**	**0**	**0.6**	**-**	×	×	
n1	× **0**	**-**	×	×	**-**	×		×
n2	× **0**	×	**-**		**-**	**-**	**-**	**-**
n3	**0.5**		×	×	**-**	**-**	**-**	**-**
n4	**0.7**	×	**-**		**-**	**-**	**-**	**-**
n5	**1.1** **0.1**	**0.2** **0.2**	**-** **0.9**	**-** **0.6**	**0**	e_{35}	**0.6**	**0**
n6	**0.6**	**-**	**1.4** **0.5**	**0.6** **0.5**	**0.1**	e_{45}	**0.2**	**0**
n7	**1.3** **0.3** **0**	**0.2** **0.2**	×	×	**0**	e_{67}	**0.4**	
n8	×	×	**0.8**	**-**	**-**	**-**		**-**
n9	**-** **0.8**	**-**	×	×	**-**	**-**	**-**	
n10	**-** × **0**	×	**1.5** **0.5**	**0.5**	**0**	e_{89}	**0.2**	
T	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**

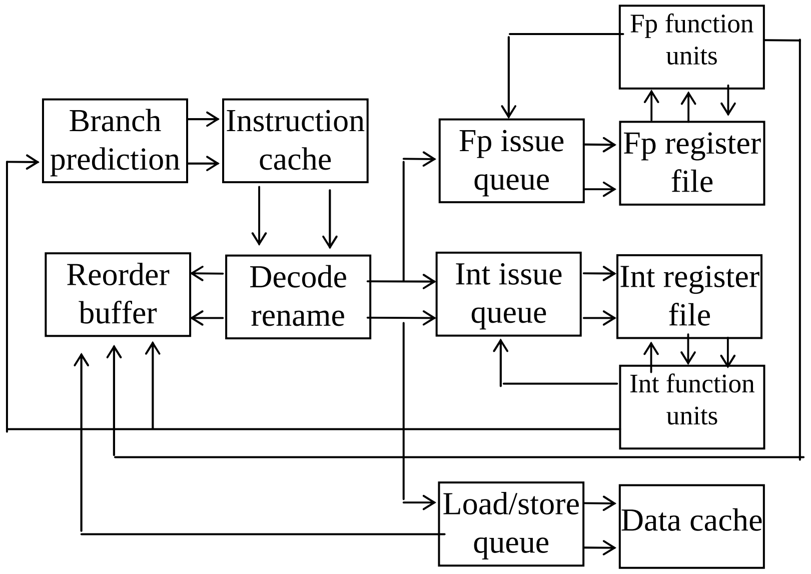
Analysis of Dynamic Scanning

- The worst complexity should be $|Path|^2|V|$.
  - $|Path|$ is the number of critical paths in the architecture and $|V|$ is the number of nodes in the graph which is about the total number of blocks and wires.
- Cycle break for sequential circuits.
  - In micro-architecture, almost every component is involved in one or more loops;
  - Break the cycles by directly changing the direction of the back edge to the target node *t;*
  - The information of the cycles will be lost, but the error is pretty small as shown in experimental results section.

Outline

Superscalar Processors

- Each of these blocks spans one or more pipeline stages, and instructions typically flow through these stages ;
- Extra delay in any of these sub-systems will have different effects on system performance.



Throughput aware floorplanning with pipelining

- **Given:**
  - Target clock period Φ
  - Clocking overhead $T_{overhead}$
  - List of blocks with their area, dimensions and delay
  - Critical architectural paths with performance sensitivities
- **Objective: Generate a floorplan which optimizes for the die area, wirelength and performance, based on the pipeline design for blocks and wires.**
  - Based on a simulated annealing framework with CBL representation
  - Integrate the graph based dynamic approach
  - Extra latency from the wires is used to compute the new BIPS

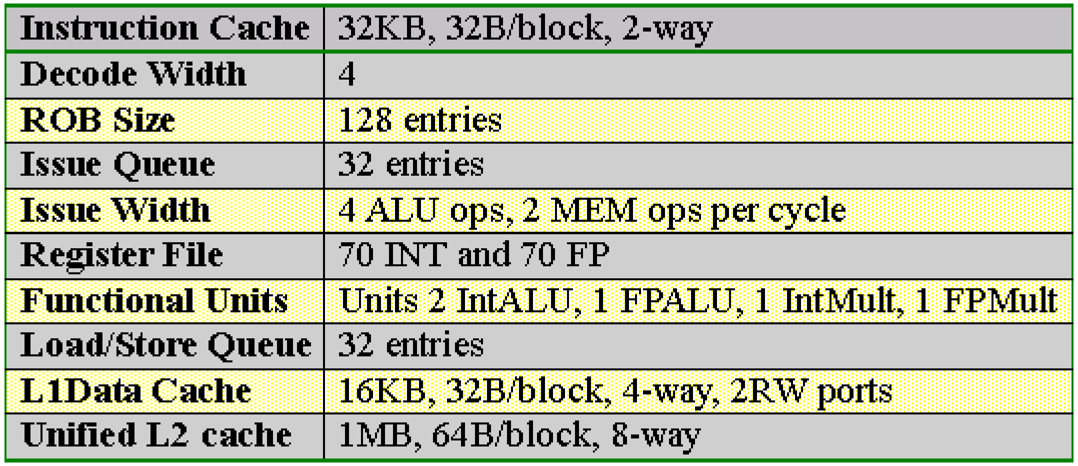
$$\cos t = w1 * \frac{1}{BIPS} + w2 * Area + w3 * Wire$$

Outline

- Introduction
- Problem Formulation
- Simultaneous Block and Interconnect Pipelining
  - MILP formulation
  - Graph-based heuristic algorithm
- Throughput aware floorplanning with pipelining
- Case Study for a Design Driver
- Conclusions and Future Works

Design Driver

- An out-of-order superscalar processor micro-architecture with 4 banks of L2 cache in 70*nm* technology.

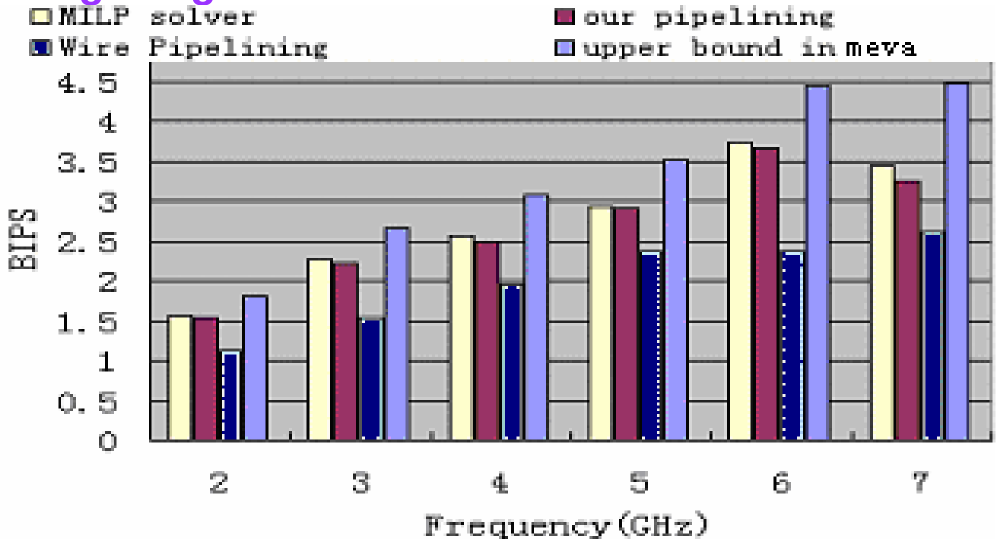


Instruction Cache	32KB, 32B/block, 2-way
Decode Width	4
ROB Size	128 entries
Issue Queue	32 entries
Issue Width	4 ALU ops, 2 MEM ops per cycle
Register File	70 INT and 70 FP
Functional Units	Units 2 IntALU, 1 FPALU, 1 IntMult, 1 FPMult
Load/Store Queue	32 entries
L1Data Cache	16KB, 32B/block, 4-way, 2RW ports
Unified L2 cache	1MB, 64B/block, 8-way

- Comparisons
  - In MEVA([Jason, DAC03 and ASPDAC05]), the flip-flops are assumed to be ideally inserted along each path, which gives an upper bound for pipeline design(UB)
  - wire-pipelining results (WP)
  - the solutions obtained from the MILP solver (MILP)
  - our graph-based heuristic approach (GH)

Impact of Frequencies

- Take a fixed packing result and run the different pipeline approaches under the frequency from 2GHz to 7GHz.
  - MEVA is far off from the real designs in most cases(on average 20% larger);
  - Wire pipelining loses a lot of the flexibility to get a better design, our approach gives about a 27% performance improvement over it;
  - Our approach has an average of 2% error to the results of the MILP;
  - The running time for our approach is about 1 second, while the MILP need about 200s to get a good result.



Different Packing results

- **Randomly pick 5 packings and run the pipelining approaches under 3GHz.**
  - **The difference between our approach and the MILP solution is about 3%;**
  - **The error for upper bound in MEVA is about 16% and the error for wire-pipelining results is about 24% .**

packing	Area (mm*mm)	wire	BIPS			
			UB	WP	GH	MILP
1	4.1*7.76	98471	2.677	1.539	2.136	2.262
2	4.24*7.92	106594	2.417	1.731	2.139	2.16
3	4.9*6.35	102578	2.547	1.748	2.091	2.202
4	5.48*5.88	134051	2.654	1.563	2.319	2.346
5	6.99*4.64	135455	2.793	1.731	2.235	2.286
Average Error			1.16	0.738	0.97	1

Integrated with floorplanning

- We integrate graph-based heuristic with the thoughput-driven floorplannning
- Take MILP approach as a post process at the end of the floorplanning.
  - It is an accurate evaluation of the pipelining design, which enables the optimization process to be guided correctly and converge well.

Frequency GHz	UB+post_MILP			GH		
	Area (mm²)	Wire (mm)	BIPS	Area (mm²)	Wire (mm)	BIPS
2	32.	115.6	1.492	31.8	142	1.714
3	34.6	103.7	2.139	33.3	108.4	2.22
4	32.4	98.7	2.776	36.1	124.3	2.828
5	32.8	126.2	2.885	32.6	94.17	3.35
6	36.0	108.4	3.636	33.7	100.3	3.882
7	35.9	112.5	3.479	36.8	129.9	3.906
Comp	1	1	1	1.003	1.05	1.091

Conclusion and Future Works

- First work that considers block pipelining and interconnect pipelining simultaneously;
  - A MILP formulation.
  - A novel *dynamic scanning heuristic*
  - heuristic gives solutions very close to the MILP results (2% more than MILP results on average) but in a much shorter runtime;
- The dynamic scanning heuristic is stable and effective which is applicable in floorplanning optimization;
- Refine the MILP formulation and attempt to handle it in a much more efficient way.

清華大學
Tsinghua University

Thank You!