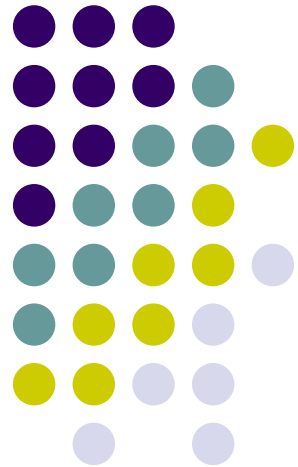
Minimizing Clock Latency Range in Robust Clock Tree Synthesis

Wen-Hao Liu
Yih-Lang Li
Hui-Chi Chen



You have to enlarge your font. Many pages are hard to view. I think the position of Page topic is too low such that the content space is limited. Replace this format by Lab slide format.
Add page number.

Outline

- Introduction
- Problem Formulation
- Algorithm Overview
- Obstacle-Avoiding Balanced Clock Tree Routing
- Monotonic Parallel BIWS
- Experimental Results
- Conclusions

Introduction

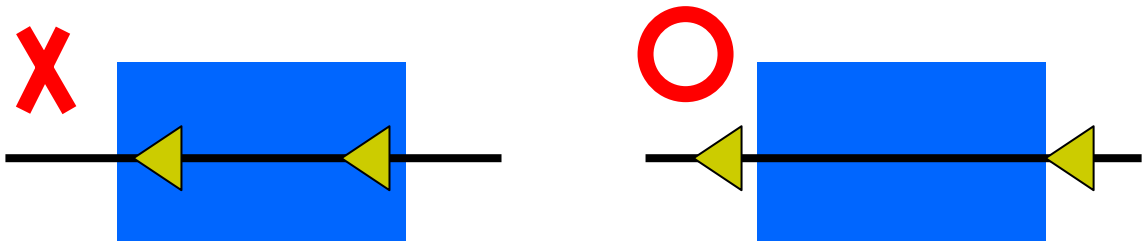
- Clock network synthesis has played an important role in determining circuit performance.
  - Skew is still one of the most important indicators.

$$clock\ period \geq t_d + t_{skew} + t_{su} + t_{ds}$$

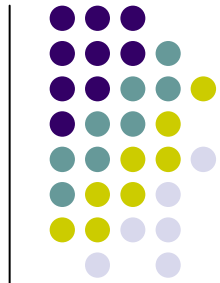
t_d:	**Longest path through combinational logic**
t_{skew}:	**Clock skew**
t_{su}:	**Setup time of the synchronizing elements**
t_{ds}:	**Propagation delay within the synchronizing element**

Introduction

- The signal degrades when propagating through a long routing wire (slew problem). Therefore, we need to insert buffers to repower the signal line so as to maintain adequate slew rate.
- Pre-placed IPs and macro blocks in a design are thought as obstacles which obstruct buffer insertion.



Introduction



- Variation of clock tree has to be concerned (Voltage variation, process variation, noise etc).
- In the ISPD'09 Clock Network Synthesis (CNS) contest, the primary goal is to minimize the clock latency range (CLR) across different supply-voltage simulations under given slew rate, obstacle, total capacitance constraints.
- To the best of our knowledge, no work has yet addressed CLR minimization.

Outline

- Introduction
- Problem Formulation
- Algorithm Overview
- Obstacle-Avoiding Balanced Clock Tree Routing
- Monotonic Parallel BIWS
- Experimental Results
- Conclusions

Problem Formulation

- ***Clock Network Synthesis* (CNS)**:
  - Given: a 8-tuple (S, SNK, OB, Ψ, W, V, Cap_{limit}, $Slew_{limit}$)
  - Objective: the CNS problem is to construct a clock tree such that $CLR(BT_S)$ is minimal.
  - Subject to:
    - $tcap(BT_S) \leq Cap_{limit}$,
    - $slew(b\_s) \leq Slew_{limit}$, $\forall b\_s \in BF(BT_S) \cup SNK$;
    - Buffers are not allowed to overlap obstacles.

S: source
SNK: a set of sinks with sink locations
OB: a set of obstacles
Ψ :a buffer library
W:a wire library
V:a set of multiple supply voltages
Cap_{limit}: total capacitance constraint
$Slew_{limit}$: slew constraint

Problem Formulation

- Clock latency Range (CLR)
  - The maximum difference of each sink's latency across different voltages
  - $CLR(BT_S) <= skew(BT_S) + MDV(BT_S)$
    - $MDV(BT_v)$ is maximum source-to-sink delay variation across multiple supply voltages.

BT	s_1	s_2
1V	15	12
1.1V	10	8

$CLR(BT_S)$ =15-8=7

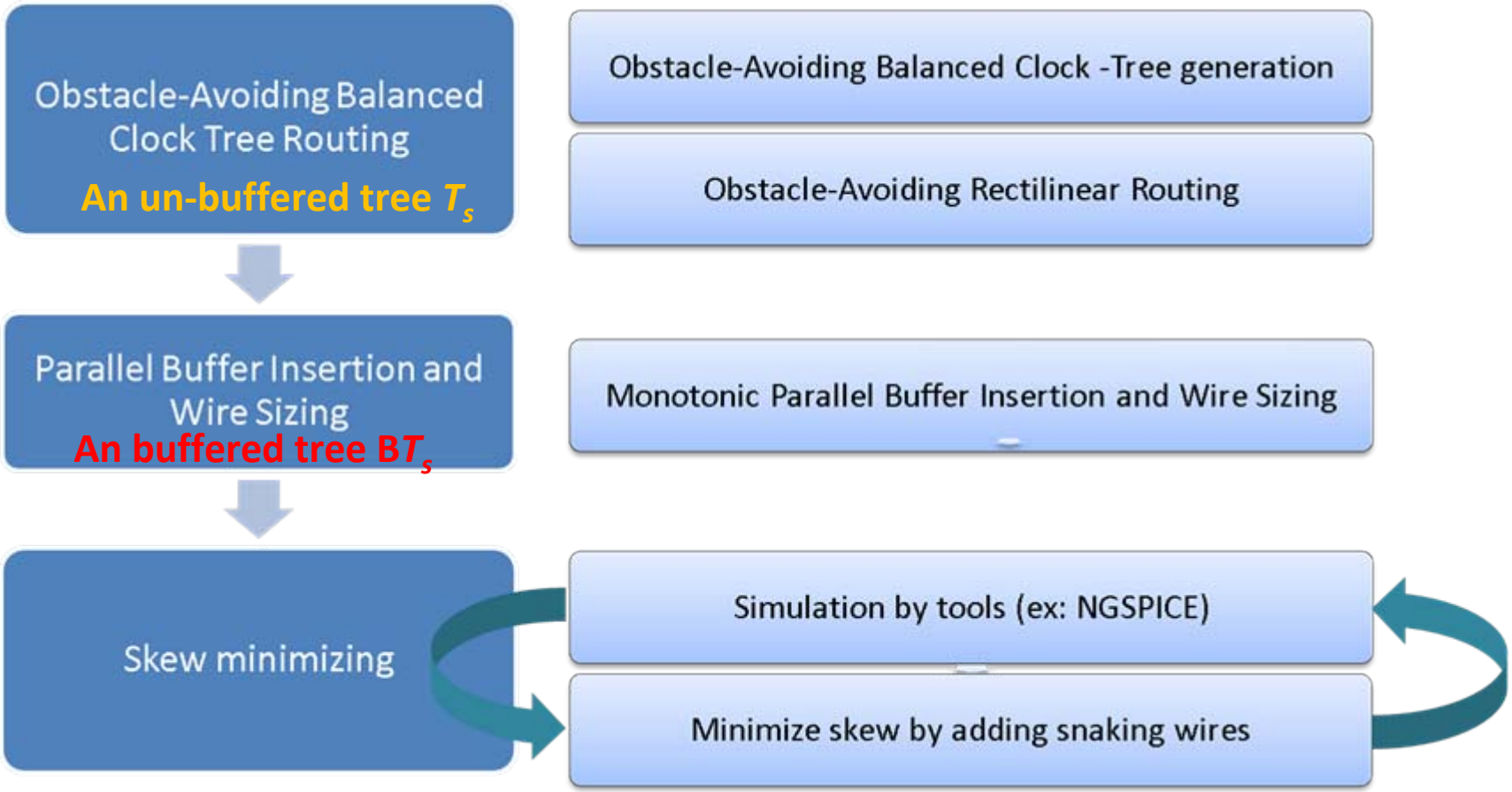
$skew(BT_S)$ =max(15-12, 10-8)=3

$MDV(BT_S)$ =max(15-10, 12-8)=5

Outline

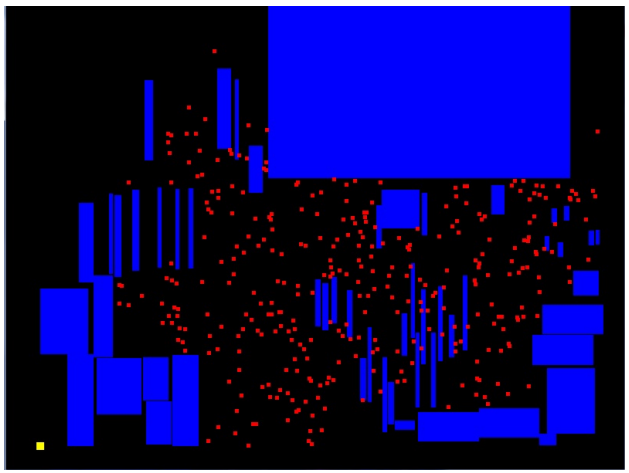
- Introduction
- Problem Formulation
- Algorithm Overview
- Obstacle-Avoiding Balanced Clock Tree Routing
- Monotonic Parallel BIWS
- Experimental Results
- Conclusions

Algorithm Flow

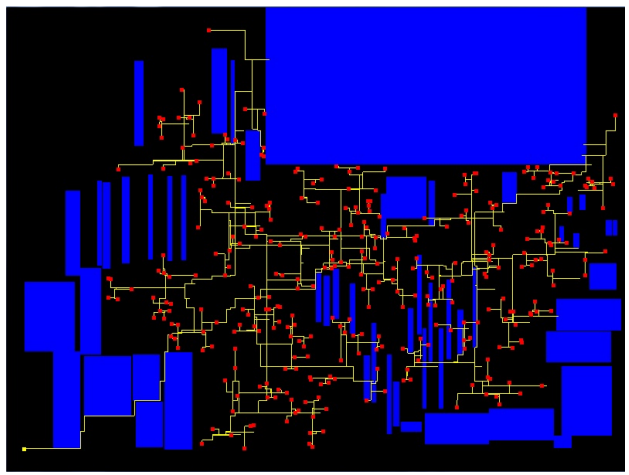


Algorithm Flow

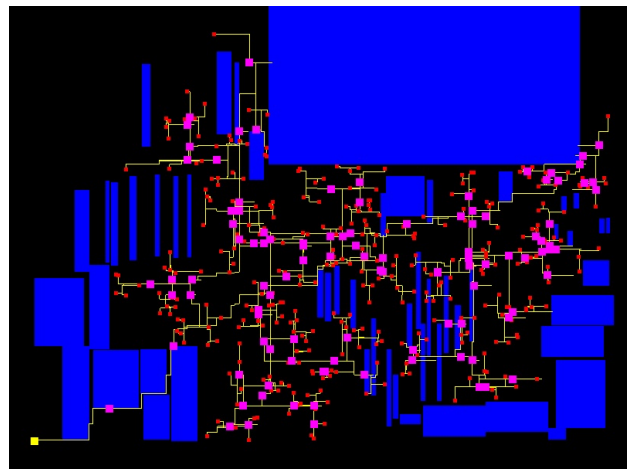
- The goal at the first stage is to identify a balanced un-buffered zero-skew clock tree *TS* with minimal wirelength.
- In the second stage, monotonic parallel BIWS algorithm inserts parallel buffers and performs wire sizing to minimize ??? under capacitance and slew constraints.
- The third stage invokes ngspice to calculate accurately the delay of each sink, and then inserts snaking wires along the path of the small delay to reduce skew.



Input

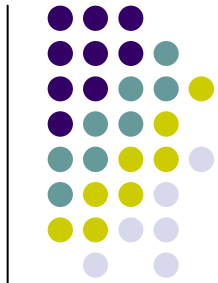


After stage1



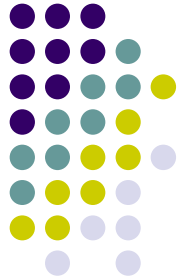
After stage2&3

Outline

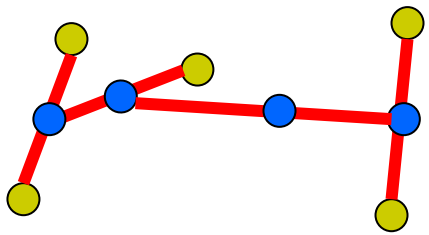


- Introduction
- Problem Formulation
- Algorithm Overview
- Obstacle-Avoiding Balanced Clock Tree Routing
  - Obstacle-Aware Balanced Clock-Tree Generation
  - Obstacle- Avoiding Rectilinear Routing
- Monotonic Parallel BIWS
- Experimental Results
- Conclusions

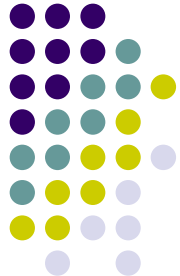
Obstacle-Aware Balanced Clock-Tree Generation



- *This approach provides tree topology and internal node location.*
  - the clustering-based method is adapted.
  - Given *n* sinks, every sink is initially regarded as an individual zero-skew sub-tree.
  - The operation merges two sub-trees as a new zero-skew sub-tree .



Obstacle-Aware Balanced Clock-Tree Generation

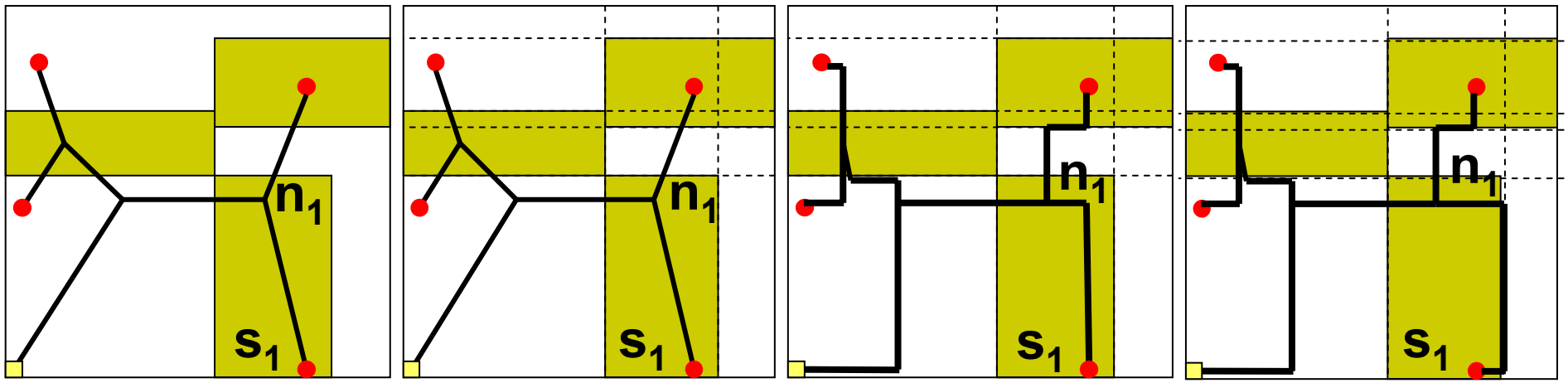


- In this work, three factors determine the candidates for merging.
  - Wirelength of connecting two sub-tree (this item is the same as the next item, do u mean “wirelength of connected two sub-trees”. It means the total wirelength of the new tree that connects two sub-trees)
  - Wirelength of the new wires that are required for tree merging, inside the obstacles
  - To generate balanced tree, we choose two sub-trees with low capacitance for merging.

$$\tau(t_1, t_2) = \alpha * wire(t_1, t_2) + \beta * ubw(t_1, t_2)^2 + \gamma * (tcap(t_1) + tcap(t_2))$$

Obstacle- Avoiding Rectilinear Routing

- 1) Construct a non-uniform grid routing graph by extending all borders of all obstacles .
- 2) Monotonic routing is adopted to seek the rectilinear routing of every tree edge.
- 3) If the length of a tree edge inside an obstacle exceeds *L*, then the maze routing is invoked to reroute this tree edge to reduce the wirelength inside an obstacle.

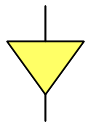
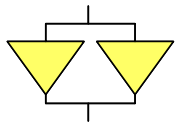
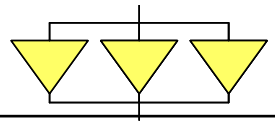
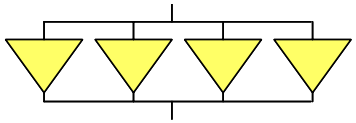


Outline

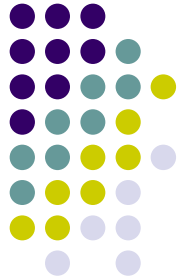
- Introduction
- Problem Formulation
- Algorithm Overview
- Obstacle-Avoiding Balanced Clock Tree Routing
- Monotonic Parallel BIWS
  - CLR-driven BIWS
  - Re-Buffered Tree Extraction using Amplified Capacitance
- Experimental Results
- Conclusions

Monotonic Parallel BIWS

- What is parallel buffer? Why use parallel buffer?

Type	Parallel buffer	Cap	intrinsic delay at 1V	intrinsic delay at 1.2V	Delay variation (DV) among 1V and 1.2V
1		115	8	6	2
2		230	7	6	1
3		345	6.67	6	0.67
4		460	6.5	6	0.5

Monotonic Parallel BIWS

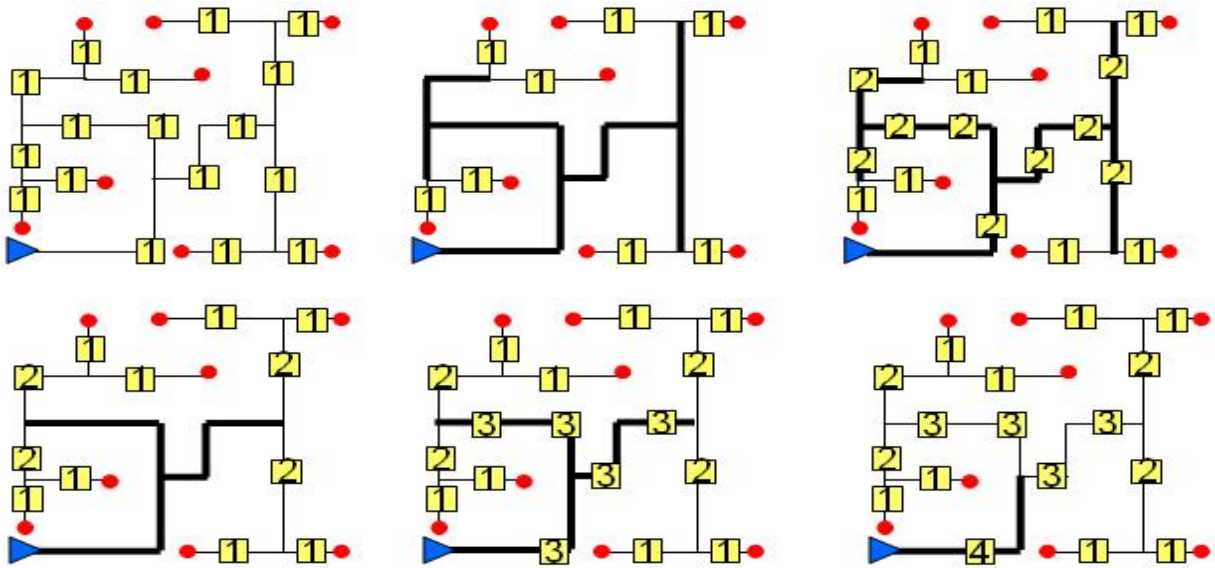


- $CLR(BT_S) \cong skew(BT_S) + MDV(BT_S)$
- To the best of our knowledge, the MDV minimization has not been investigated. In monotonic parallel BIWS, parallel buffers are exploited to decline MDV.
- Traditional BIWS algorithms [4][5][6][8] have to search a large solution space if the buffer library becomes large.

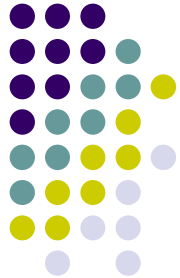
The font is too small. Try to reduce the word count. The position of The topic is too low such that the content space is reduced.

Monotonic Parallel BIWS

- The proposed buffer insertion strategy is to place the least-DV parallel buffer as close as possible to the source, to reduce MDV of every path from source to every sink.
- In this work the solution space does not increase markedly because of using single type of parallel buffers in each re-buffering iteration.



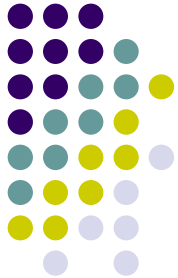
The font is too small. Try to reduce the word count. The position of The topic is too low such that the content space is reduced.



CLR-Driven BIWS

- Given an un-buffered tree, CLR-driven BIWS seeks out a minimum *weighted-CLR* buffered clock tree. The *weighted-CLR* is defined as follows:
  - $WCLR\,(BTS) = skew(BTS) + \omega \times MDV(BTS),\ \omega > 1$
- CLR-driven BIWS is a Ginneken-like algorithm that enumerates all possible BIWS solutions and then selects the minimum weighted-CLR solution from all solutions.
- Pruning redundant solutions is a principal issue in designing an efficient Ginneken-like algorithm.
  - *Look-ahead violation-detecting scheme*
  - *Grouping and pruning suboptimal solutions*
  - *Pruning surplus solutions*

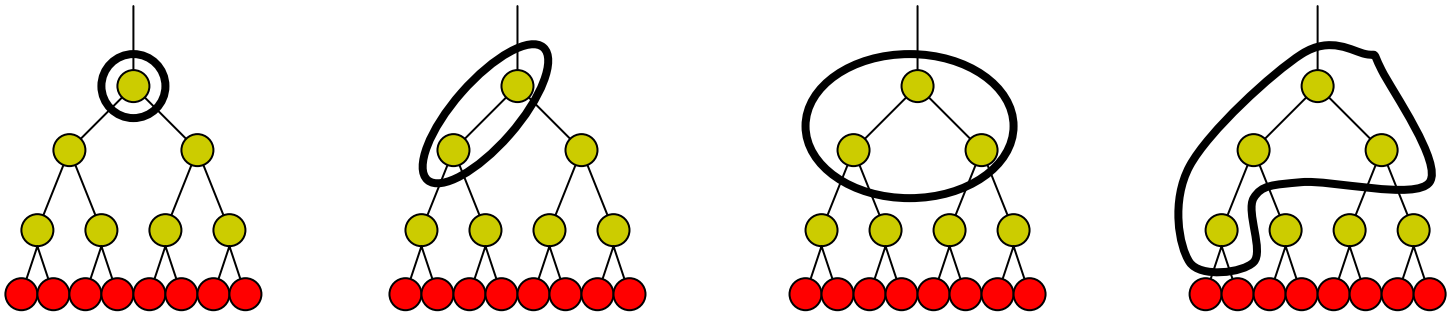
Re-Buffered Tree Extraction using Amplified Capacitance



Algorithm *GetReBufTree*

Input: Clock tree $BT_S(i\text{-}1)$, buffer b_i ;

1. $RPBT_S(i) \leftarrow \varnothing$; $C_{orig} \leftarrow tcap(BT_S(i-1))$; $C_{incr} \leftarrow 0$;
2. **while** ($RPBT_S(i)$!= $BT_S(i-1)$)
3. $e \leftarrow$ Breadth_First_Visit_Tree ($BT_S(i-1)$);
4. $C_{incr} = C_{incr} + reBufIncCap(e, b_i)$;
5. **if** ($C_{orig} + \lambda \times C_{incr} < Cap_{limit}$)
6. $RPBT_S(i) = RPBT_S(i) + e$;
7. **else** **break** ;
8. **return** $RPBT_S(i)$;



Outline

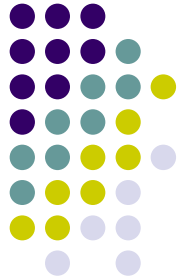
- Introduction
- Problem Formulation
- Algorithm Overview
- Obstacle-Avoiding Balanced Clock Tree Routing
- Monotonic Parallel BIWS
- Experimental Results
- Conclusions

Experimental Environment

- ISPD09 benchmarks were used.
- Machine: Intel Xeon 3.0GHz CPU and 16GB RAM.
- Runtime is normalized according to the clock rate ratio.

The template of this page is different from the other.

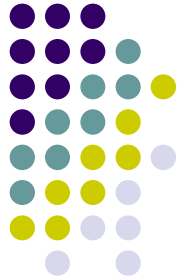
Routing Result Statistics



	[5] + CLR-driven BIWS				Our stage1 + CLR-driven BIWS			
name	CLR	Max Skew	Total CAP	CPU(s)	CLR	Max Skew	Total CAP	CPU(s)
ispd09f11	85.26	35.99	59224.61	55.49	67.51	19.38	56184.99	64.09
ispd09f12	75.06	29	53038.03	55.92	68.97	21.47	51860.92	58.91
ispd09f21	81.15	30.23	60819.47	52.46	74.95	24.06	60104.27	60.4
ispd09f22	61.39	24.1	37594.23	35.65	60.67	24.06	36618.17	39.88
ispd09f31	X	X	X	X	112.77	37.24	124917.11	188.42
ispd09f32	X	X	X	X	94.8	23.64	92935.77	122.53
ispd09fnb1	X	X	X	X	43.58	28.92	21679.2	697.53

	Our stage1 + stage2				Our stage1+stage2+stage3			
name	CLR	Max Skew	Total CAP	CPU(s)	CLR	Max Skew	Total CAP	CPU(s)
ispd09f11	33.39	20.18	101500.14	469.19	18.77	7.12	102144	2200
ispd09f12	26.44	12.64	96248.01	446.8	15.5	3.06	96715	1923
ispd09f21	28.4	13.52	110458.77	468.51	17.04	3.02	111094	2231
ispd09f22	30.27	19.24	69016.88	342.67	16.25	4.11	69425	1370
ispd09f31	50.72	36.33	221743.08	974.37	22.63	7.58	223337	6360
ispd09f32	36.81	21.4	168294.07	712.19	20.59	5.52	169204	3967
ispd09fnb1	25.4	34.53	37753.24	1119.26	14.32	3.77	38782	1743

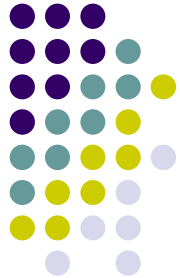
Routing Result Statistics



	[5] + CLR-driven BIWS				Our stage1 + CLR-driven BIWS			
name	CLR	Max Skew	Total CAP	CPU(s)	CLR	Max Skew	Total CAP	CPU(s)
ispd09f11	85.26	35.99	59224.61	55.49	67.51	19.38	56184.99	64.09
ispd09f12	75.06	29	53038.03	55.92	68.97	21.47	51860.92	58.91
ispd09f21	81.15	30.23	60819.47	52.46	74.95	24.06	60104.27	60.4
ispd09f22	61.39	24.1	37594.23	35.65	60.67	24.06	36618.17	39.88
ispd09f31	X	X	X	X	112.77	37.24	124917.11	188.42
ispd09f32	X	X	X	X	94.8	23.64	92935.77	122.53
ispd09fnb1	X	X	X	X	43.58	28.92	21679.2	697.53

	Our stage1 + stage2				Our stage1+stage2+stage3			
name	CLR	Max Skew	Total CAP	CPU(s)	CLR	Max Skew	Total CAP	CPU(s)
ispd09f11	33.39	20.18	101500.14	469.19	18.77	7.12	102144	2200
ispd09f12	26.44	12.64	96248.01	446.8	15.5	3.06	96715	1923
ispd09f21	28.4	13.52	110458.77	468.51	17.04	3.02	111094	2231
ispd09f22	30.27	19.24	69016.88	342.67	16.25	4.11	69425	1370
ispd09f31	50.72	36.33	221743.08	974.37	22.63	7.58	223337	6360
ispd09f32	36.81	21.4	168294.07	712.19	20.59	5.52	169204	3967
ispd09fnb1	25.4	34.53	37753.24	1119.26	14.32	3.77	38782	1743

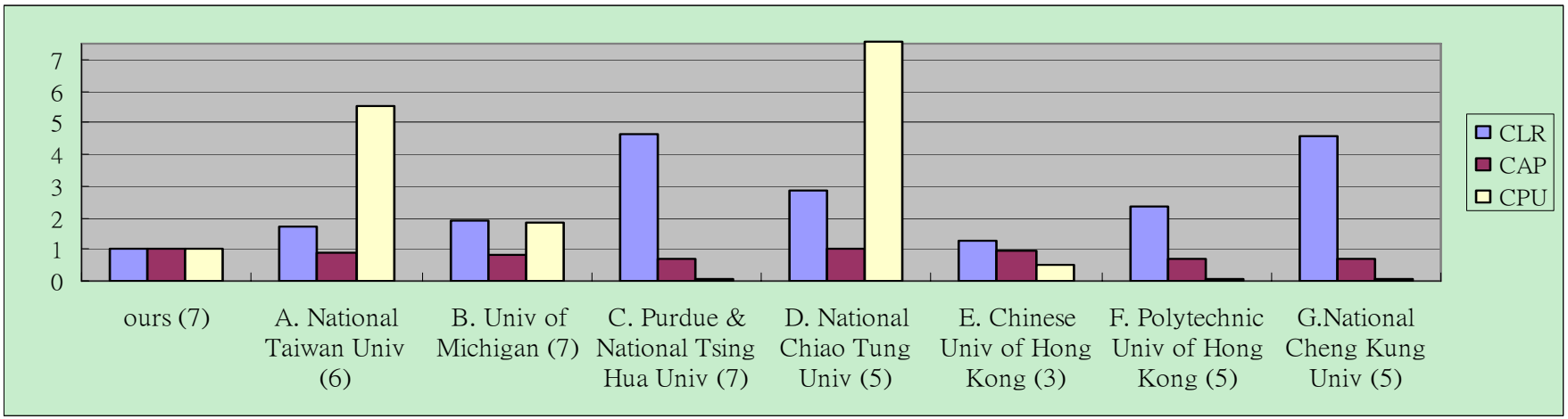
Routing Result Statistics



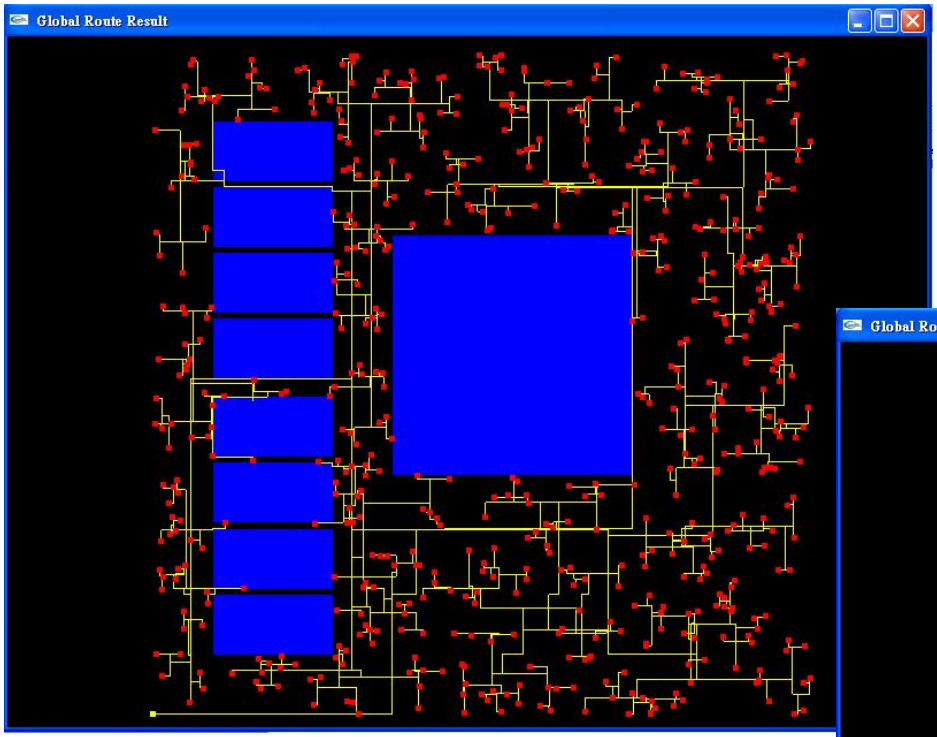
	[5] + CLR-driven BIWS				Our stage1 + CLR-driven BIWS			
name	CLR	Max Skew	Total CAP	CPU(s)	CLR	Max Skew	Total CAP	CPU(s)
ispd09f11	85.26	35.99	59224.61	55.49	67.51	19.38	56184.99	64.09
ispd09f12	75.06	29	53038.03	55.92	68.97	21.47	51860.92	58.91
ispd09f21	81.15	30.23	60819.47	52.46	74.95	24.06	60104.27	60.4
ispd09f22	61.39	24.1	37594.23	35.65	60.67	24.06	36618.17	39.88
ispd09f31	X	X	X	X	112.77	37.24	124917.11	188.42
ispd09f32	X	X	X	X	94.8	23.64	92935.77	122.53
ispd09fnb1	X	X	X	X	43.58	28.92	21679.2	697.53

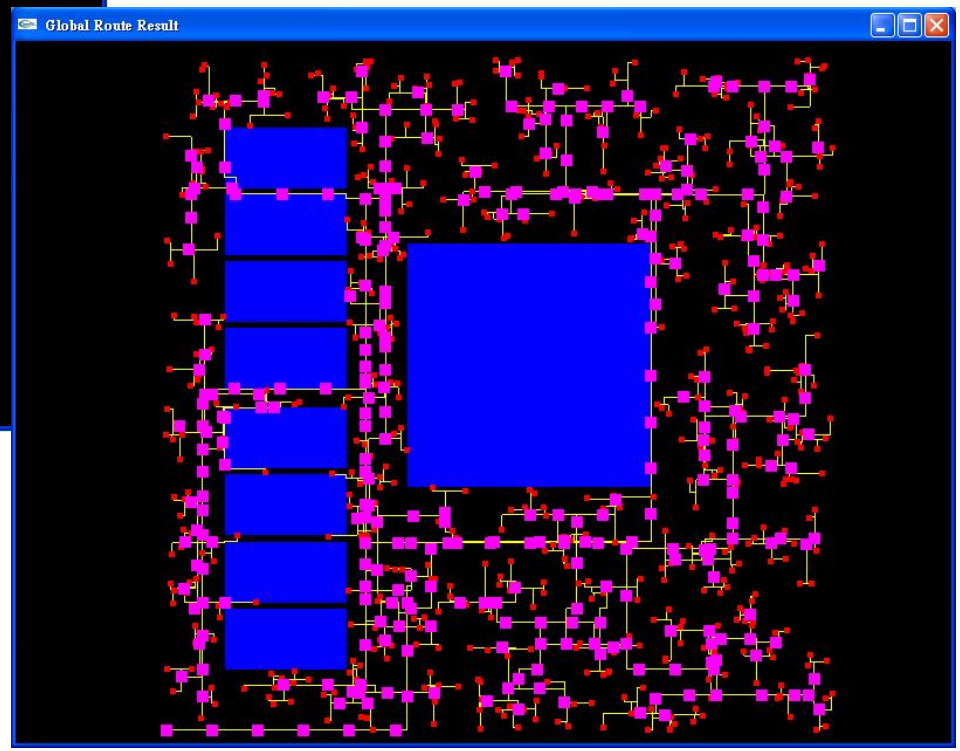
	Our stage1 + stage2				Our stage1+stage2+stage3			
name	CLR	Max Skew	Total CAP	CPU(s)	CLR	Max Skew	Total CAP	CPU(s)
ispd09f11	33.39	20.18	101500.14	469.19	18.77	7.12	102144	2200
ispd09f12	26.44	12.64	96248.01	446.8	15.5	3.06	96715	1923
ispd09f21	28.4	13.52	110458.77	468.51	17.04	3.02	111094	2231
ispd09f22	30.27	19.24	69016.88	342.67	16.25	4.11	69425	1370
ispd09f31	50.72	36.33	221743.08	974.37	22.63	7.58	223337	6360
ispd09f32	36.81	21.4	168294.07	712.19	20.59	5.52	169204	3967
ispd09fnb1	25.4	34.53	37753.24	1119.26	14.32	3.77	38782	1743

ISPD'09 Clock Network Synthesis Contest



Synthesis result





Outline

- Introduction
- Problem Formulation
- Algorithm Overview
- Obstacle-Avoiding Balanced Clock Tree Routing
- Monotonic Parallel BIWS
- Experimental Results
- Conclusions

Conclusions

- In this work, a three-stage CLR-driven CTS flow is proposed.
- An obstacle-avoiding balanced clock tree routing algorithm identifies a balanced clock tree with minimal wirelength to minimize clock skew and wire capacitance.
- An efficient monotonic parallel BIWS algorithm is presented to a minimum-weighted-CLR buffered clock tree.
- The proposed CLR-driven CTS flow can complete all ISPD'09 benchmark circuits and yield 59%, 52.7% and 35.4% lower CLRs than those produced by the top three performers of the ISPD'09 CNS contest.