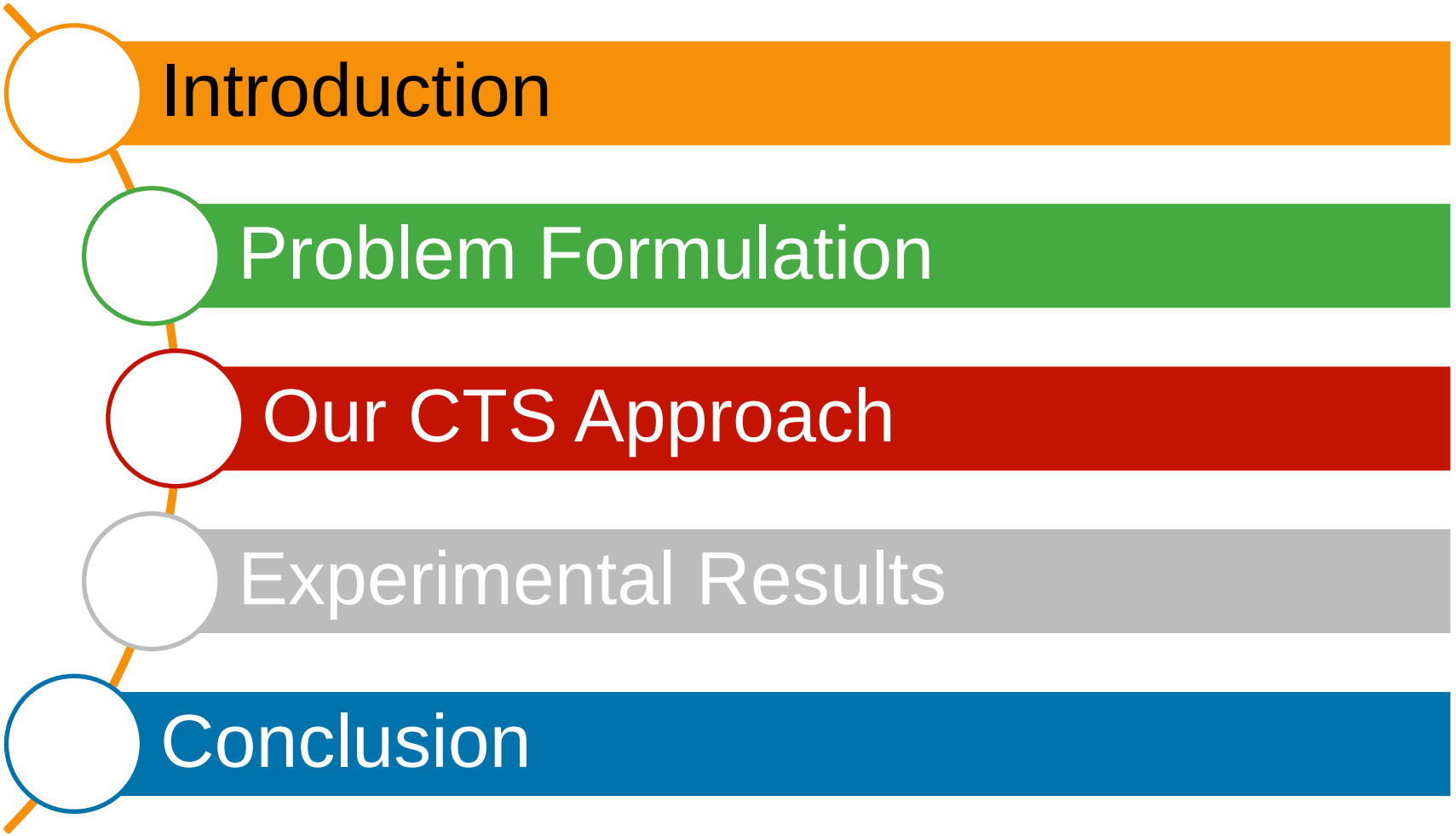
Clock Design Methodology for Energy and Computation Efficient Bitcoin Mining Machines

Chien-Pang Lu, Iris Hui-Ru Jiang, Chih-Wen Yang

國立臺灣大學 National Taiwan University

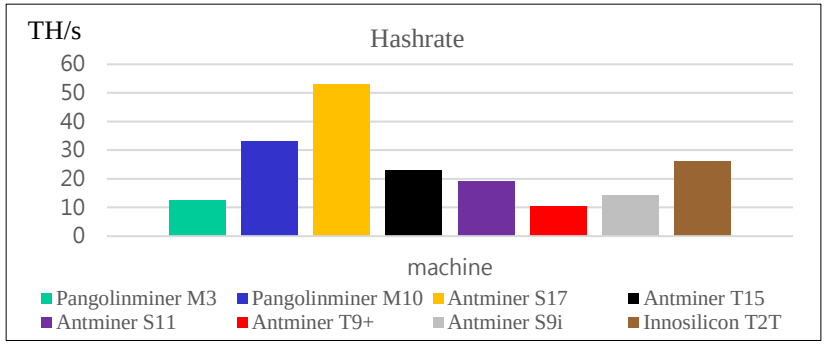
DIGWISE TECHNOLOGY 崛智科技有限公司

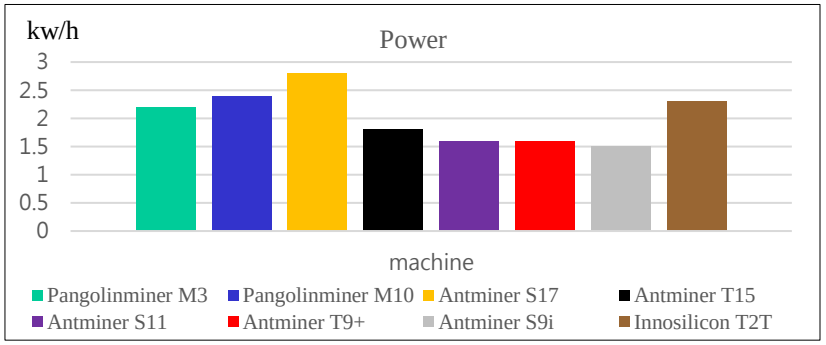
Outline


Introduction
Problem Formulation
Our CTS Approach
Experimental Results
Conclusion

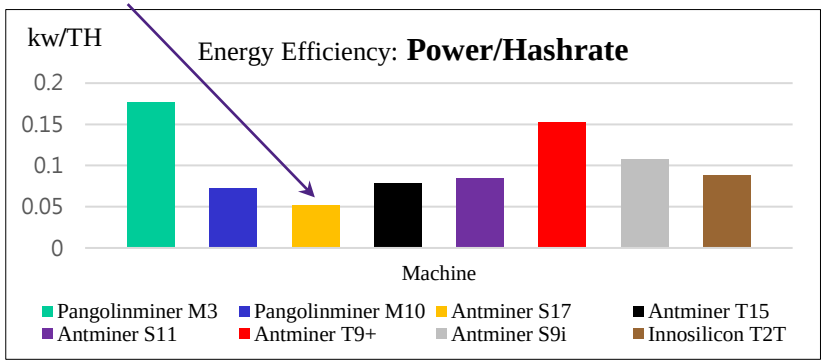
Energy Consumption & Throughput

- Mining machine throughput(TH: tera hash) & power compare





Target: low “energy efficiency” → high CP value

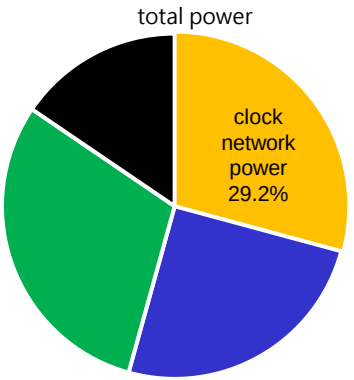


Pangolinminer M3: 28nm
Pangolinminer M10: 16nm
Antiminer S17: 7nm
Antiminer T15: 7nm
Antiminer S11: 16nm
Antiminer T9+: 16nm
Antiminer S9i: 16nm
Innosilicon T2T: 10nm
cite: http://www.rhy.com/en_us/rental

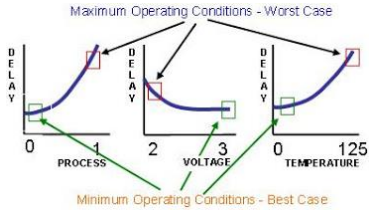
Energy Consumption & Throughput

- Maximum efficient energy ⇔ minimum money cost
  - Target: Power / Hashrate↓
    - Power($P = C_{clk}V_{dd}^2 f_{clk} + P_S$)↓
      - Reduce device count
      - v_{dd}: device internal working voltage
        - Reduce v_{dd} (use low-V device)
      - C_{clk}: clock switching cap. charged/discharged
        - Reduce network capacitance
      - f_{clk}: clock frequency
        - Reduce clock frequency
    - Hashrate(throughput)↑
      - Increase pipe depth
      - Increase f_{clk}
      - Operating condition PVT
        - Increase v_{dd}
        - Decrease T

contradiction



$$f_{clk} \propto \frac{(v_{dd} - v_{th})^{\alpha}}{V_{dd}}$$



V_{dd} is now only slightly larger than V_{th}
2013 technology(45nm and below) not only is α more like 1.3 than 2
Kyung, Yoo et al. Energy-Aware System Design Algorithms and Architectures

$P_{fixed=1}VT_{fixed=50^oc}$: Max Efficient Cost

- SHA-256 block diagram
  - Deep pipeline structure(64 stages)
    - Unroll stages to get maximum throughput

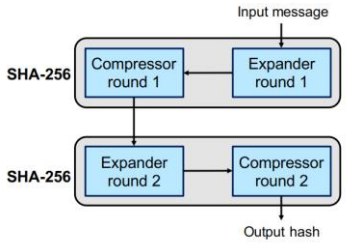


Figure 3: Block diagram of the hashing in bitcoin mining, containing two rounds of SHA-256 in series.

- Optimal f_{clk} & PVT for best efficient energy cost
  - Cost function: power * area / frequency
    - Target: search the lowest cost PVT
  - Minimum cost PVT: not the maximum frequency
    - Trick/idea: use possible setup time margin for hold area?

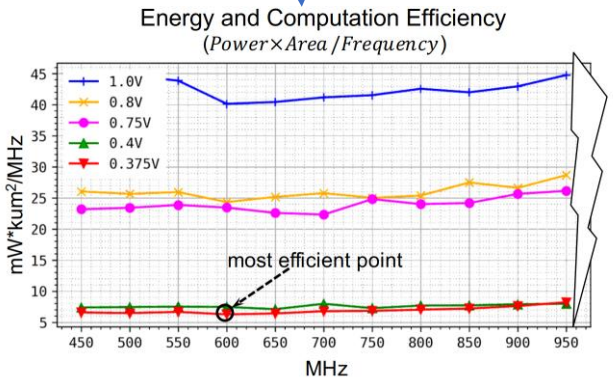


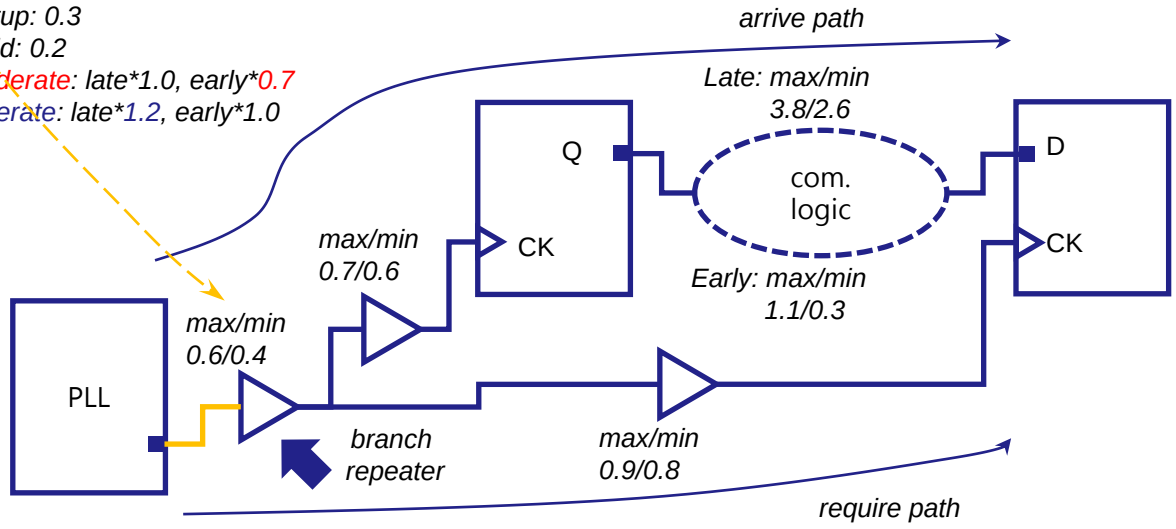
Figure 1: Power (P), area (A), frequency (F) efficiencies under different supply voltage (V) and frequency (F) settings. (Temperature: $50^{\circ}C$, TSMC 7 nm process.)

Advanced Node OCV Effect

CRPR: Clock Re-convergence Pessimism Removal

- Clock tree early branch induce OCV
- Longer CRPR path, less OCV effect
  - If possible, postpone branch clock buffer

Ex(derate on require path):
period: 4.5
lib_setup: 0.3
lib_hold: 0.2
*setup derate: late*1.0, early*0.7*
*hold derate: late*1.2, early*1.0*



$Terminology \Rightarrow$
$arrT$: *arrival time*
$reqT$: *require time*
com_{crpr}: *CRPR time*
$slack$: *slack time*

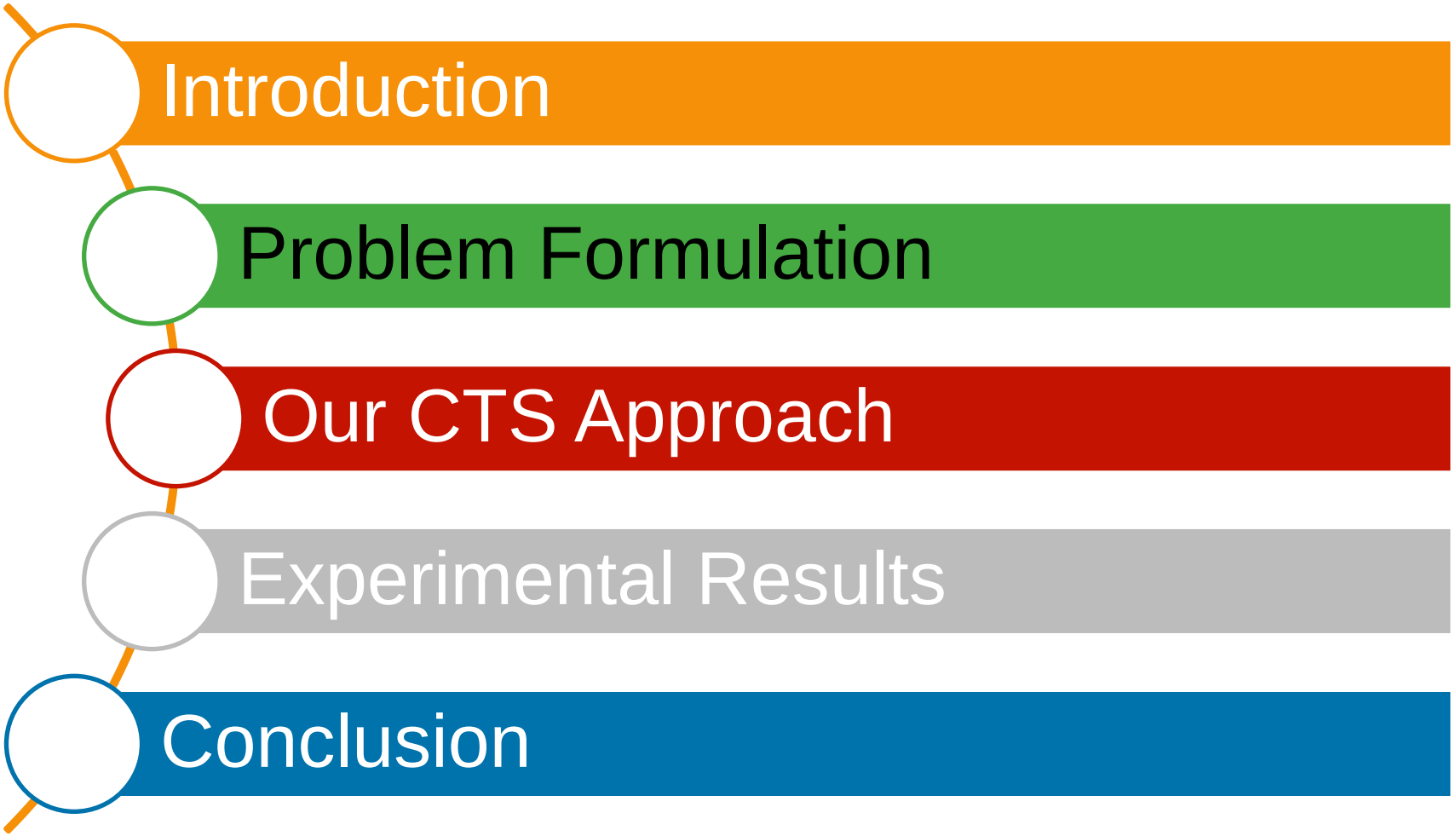
$Setup\ Check(req - arr + CRPR - lib\_setup) \Rightarrow$
$arrT_{max}: (0.6 + 0.7 + 3.8) \times 1.0 = 5.1$
$reqT_{max}: (0.6 + 0.9) \times 0.7 + 4.5 = 5.55$
$com_{crpr}: 0.6 \times (1.0 - 0.7) = 0.18$
$slack: 5.55 - 5.1 + 0.18 - 0.3 = 0.33$

note1:
useful skew
compensation: step ↔ hold

$Hold\ Check(arr - req + CRPR - lib\_hold) \Rightarrow$
$arrT_{min}: (0.4 + 0.6 + 0.3) \times 1.0 = 1.3$
$reqT_{min}: (0.4 + 0.8) \times 1.2 = 1.44$
$com_{crpr}: 0.4 \times (1.2 - 1.0) = 0.08$
$slack: 1.3 - 1.44 + 0.08 - 0.2 = -0.26$

note2:
extend CRPR as long as possible!

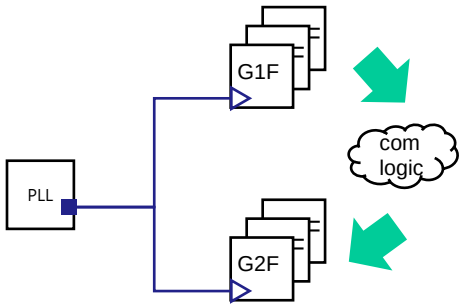
Outline


Introduction
Problem Formulation
Our CTS Approach
Experimental Results
Conclusion

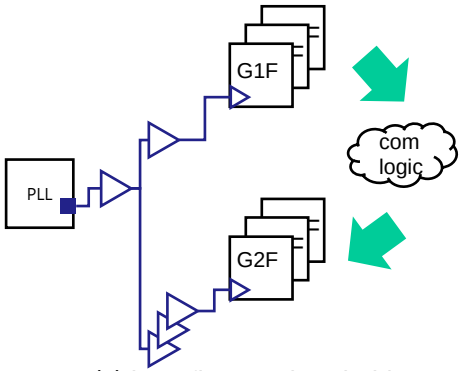
Problem Formulation

- The Timing Window Net Buffering Problem
  - Given
    - Placement location, slew constraint, buffer library, data flow & net connection
  - Goal
    - Construct timing stage sequence
    - Minimize clock branches between two stages
    - Satisfy slew constraint
    - Balance stage sinks with delay insertion

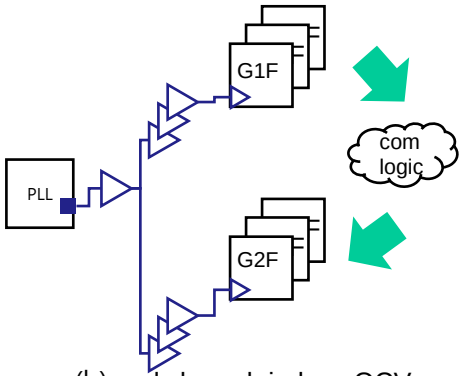
Traditional Clock Type



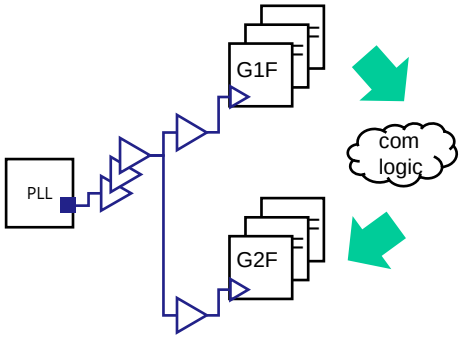
clock net connection



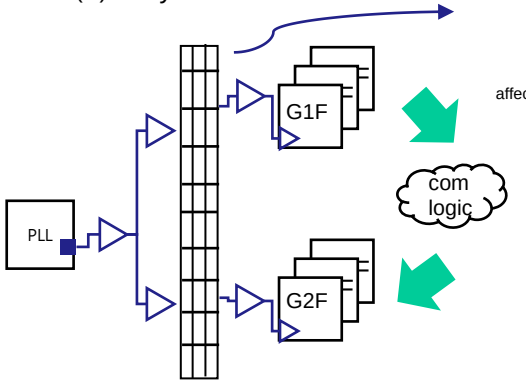
(a) benefit setup hurt hold



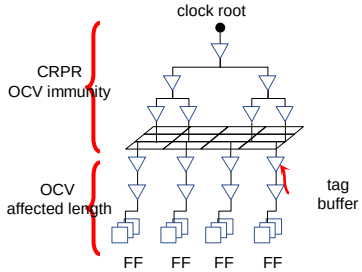
(b) early branch induce OCV



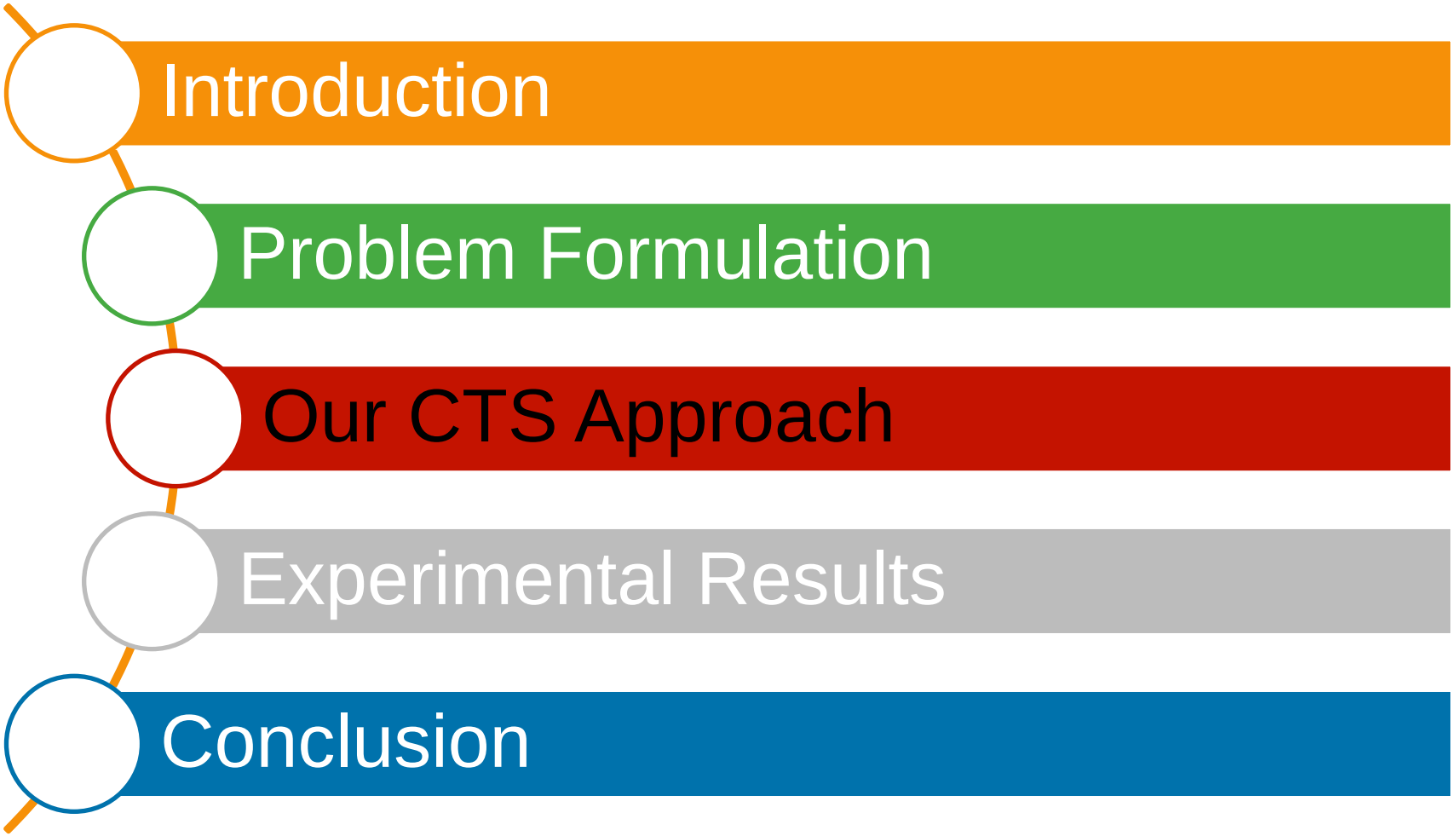
(c) less OCV/setup/hold



(d) clock mesh type



Outline


Introduction
Problem Formulation
Our CTS Approach
Experimental Results
Conclusion

Propose Clock Planning

- Bitcoin SHA-256 simple conclude
  - Plug-in application
  - Unrolled deep pipeline
  - Advanced node OCV issue, long CRPR requirement
    - Global skew or local skew?
    - Reduce clock power – less CTS buffer, less hold buffer
- Clock reversing tree
  - Useful skew setup for hold
    - Boundary scan

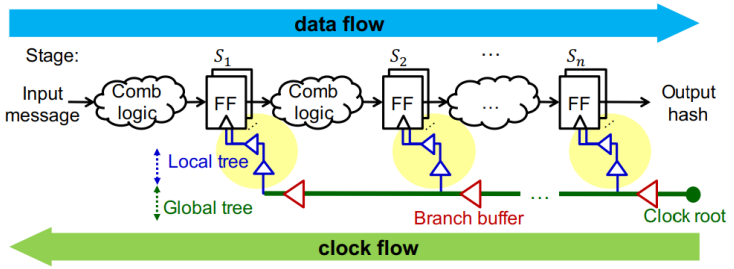
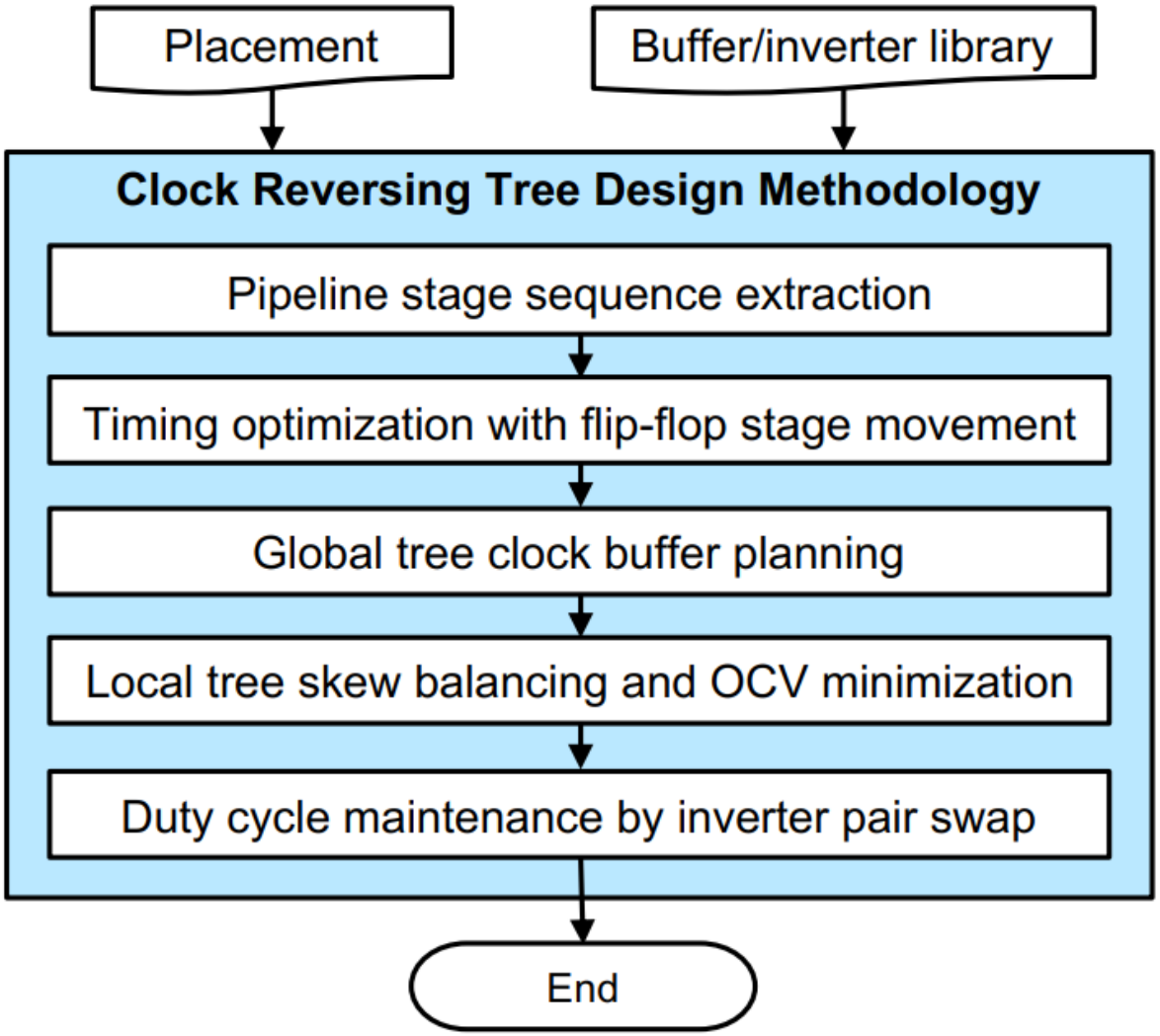


Figure 2: Clock reversing tree: Data flow in a mining machine is in the opposite direction of clock propagation.

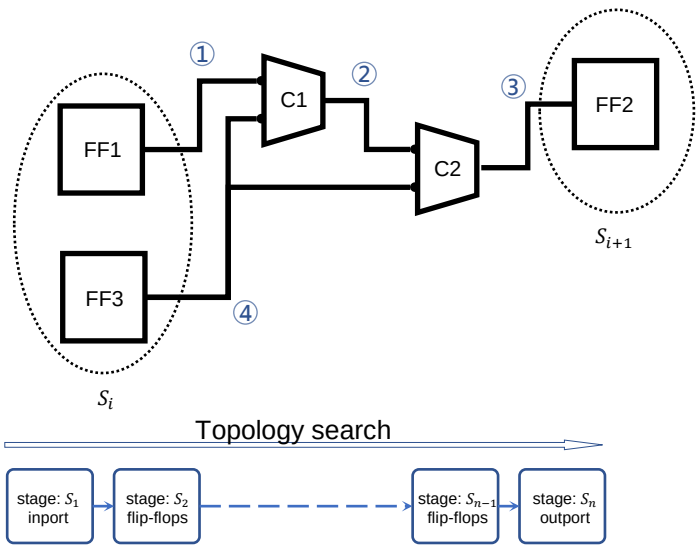
Main Flow


Placement
Buffer/inverter library
Clock Reversing Tree Design Methodology
Pipeline stage sequence extraction
Timing optimization with flip-flop stage movement
Global tree clock buffer planning
Local tree skew balancing and OCV minimization
Duty cycle maintenance by inverter pair swap
End

Pipeline Stage Sequence Extraction

Topology netlist path stamping, O(n) node traversal, n is # net

- Early timing prediction on violation path
  - Build up clock buffer hint
    - source: inport/FF outpin
    - sink: FF inpin/outport



step1(trace):
FF1 → C1
step2(trace):
C1 → C2
step3(trace & back-annotate):
C2 → FF2
stamp C2 to FF2: (min:0 , max:0)
stamp C1 to FF2: (min:1 , max:1)
stamp FF1 to FF2: (min:2 , max:2)
step4(trace & annotate):
FF3 → C1, reached
stamp FF3 to FF2: (min:2 , max:2)
FF3 → C2, reached
stamp FF3 to FF2: (min:1 , max:2)
Result:
Single/same source path path view:
FF1 → FF2 (min:2 , max:2)
FF3 → FF2 (min:1 , max:2)
Stage path view:
S_i (min:1 , max:2)

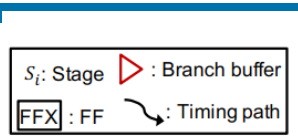
Timing Optimization with Flip-flop Movement

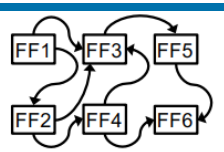
- Early timing analysis to check if flip-flops satisfy the timing constraint under the pipeline stage sequence assignment
- Terminology: $stage: S_i$
  $maximum, minimum\ delay: D_{max}, D_{min}$
  $clock\ branch\ buffer\ delay:\ D_{cbb}$
  $flip\ flop\ setup\ time:\ t_s$
  $flip\ flop\ hold\ time:\ t_h$
  $clock\ period:\ T$
  $setup\ time\ slack:\ W_s(i,j) = T - t_s - D_{max} - (j-i)D_{cbb}$
  $hold\ time\ slack:\ W_h(i,j) = D_{min} - t_h + (j-i)D_{cbb}$
- Objective function: $W_s(i,j) \geq 0\ ,\ W_h(i,j) \geq 0$
- Target candidate: $W_h(i,j) < 0\ ,\ W_s(i,j) + W_h(i,j) \geq 0$

Global Tree Clock Buffer Planning

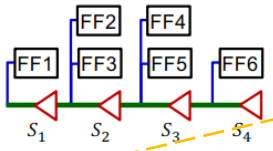
Flipflop stage optimization

- Iterative flip-flop movement optimization
  - Table look-up, search repair
- Ex(unit delay, setup/hold slack opt.):
  - Clock branch buffer delay $D_{cbb} = 1$
  - Setup time constraint $T - t_s = 9$
    - $setup\ time\ slack\ formula$: $W_s(i,j) = T - t_s - D_{max} - (j-i)D_{cbb}$
    - Ex(path view FF1 → FF2, $D_{max} = 7$, $D_{cbb} = 1$), setup time: 9 – 7 – 1 = 1
  - Hold time constraint $t_h = 3$
    - $hold\ time\ slack$: $W_h(i,j) = D_{min} - t_h + (j-i)D_{cbb}$
    - Ex(path view FF3 → FF5, $D_{min} = 5$, $D_{cbb} = 1$), hold time: 5 – 3 + 1 = 3
  - Figure:
    - (a)Initial pipeline stage sequence extraction
      - Construct FF stage max/min path delay relation table
    - (b)Moving FF5 to stage S_1
      - Check stage slack and move FF to reduce path slack
    - (c)Moving FF3 to stage S_3
      - Check stage slack and move FF to reduce path slack



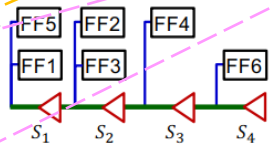


Setup time constraint: 9
Hold time constraint: 3
Branch buffer delay: 1



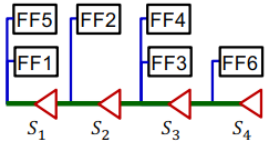
setup (WS/TS): (0/9)
hold (WS/TS): (-2/5)
(a)

Single path view			Same source view		Stage path view	
from	to	max/min: setup/hold	from	max/min: setup/hold	stage	max/min: setup/hold
FF1	FF2	7/4:+1/+2	FF1	7/2:+1/0	S_1	7/2:+1/0
FF1	FF3	6/2:+2/0				
FF2	FF3	8/4:+1/+1	FF2	8/3:+1/+1	S_2	8/3:0/+1
FF2	FF4	7/3:+1/+1				
FF3	FF5	8/5:0/+3	FF3	8/3:0/+3		
FF4	FF3	9/3:+1/-1	FF4	9/3:0/-1	S_3	9/0:0/-2
FF4	FF6	8/3:0/+1				
FF5	FF6	5/0:+3/-2	FF5	5/0:+3/-2		



setup (WS/TS): (0/9)
hold (WS/TS): (-1/5)
(b)

Single path view			Same source view		Stage path view	
from	to	max/min: setup/hold	from	max/min: setup/hold	stage	max/min: setup/hold
FF1	FF2	7/4:+1/+2	FF1	7/2:+1/0	S_1	7/2:+1/0
FF1	FF3	6/2:+2/0				
FF5	FF6	5/0:+1/0	FF5	5/0:+1/0		
FF2	FF3	8/4:+1/+1	FF2	8/3:+1/+1	S_2	8/3:+1/+1
FF2	FF4	7/3:+1/+1				
FF3	FF5	8/5:+2/+1	FF3	8/5:+2/+1		
FF4	FF3	9/3:+1/-1	FF4	9/3:0/-1	S_3	9/0:0/-1
FF4	FF6	8/3:0/+1				



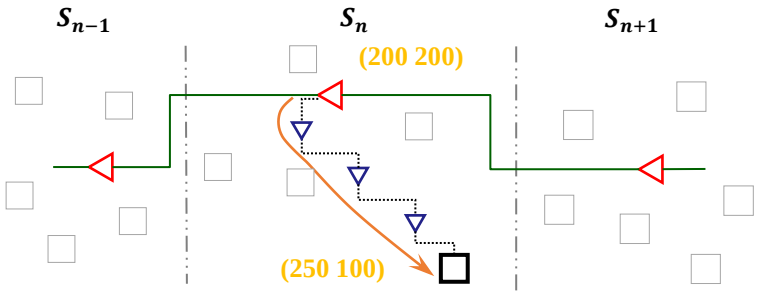
setup (WS/TS): (0/9)
hold (WS/TS): (0/5)
(c)

Single path view			Same source view		Stage path view	
from	to	max/min: setup/hold	from	max/min: setup/hold	stage	max/min: setup/hold
FF1	FF2	7/4:+1/+2	FF1	7/2:+1/0	S_1	7/2:+1/0
FF1	FF3	6/2:+2/0				
FF5	FF6	5/0:+1/0	FF5	5/0:+1/0		
FF2	FF3	8/4:+1/+1	FF2	8/3:+1/+1	S_2	8/3:+1/+1
FF2	FF4	7/3:+1/+1				
FF3	FF5	8/5:+3/0	FF3	8/5:+3/0		
FF4	FF3	9/3:0/0	FF4	9/3:0/0	S_3	9/0:0/0
FF4	FF6	8/3:0/+1				

Global Tree Clock Buffer Planning Cont.

Branch/slew/hold buffer insertion

- Branch & slew buffer: truck insertion



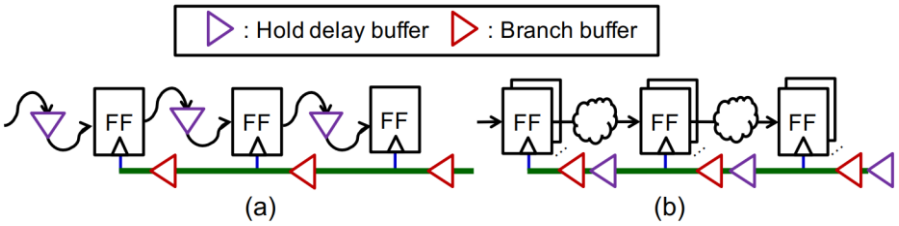
S_i :stage
:branch buffer (strength: 50um)
:slew buffer (strength: 50um)
:sink node
:farthest sink node (to branch 150um)
GMSBL: 3

GMSBL: Global Maximum Slew Buffer Level
⇒ farthest sink to branch

Ex:
global maximum DRV distance((250-200) +(200-100)) : 150um
buffer driving strength(under DRV constraint) : 50um
GMSBL(150 / 50): 3

Cite: COSAT: Congestion, Obstacle, and Slew Aware Tree Construction for Multiple Power Domain Design, DAC 2018

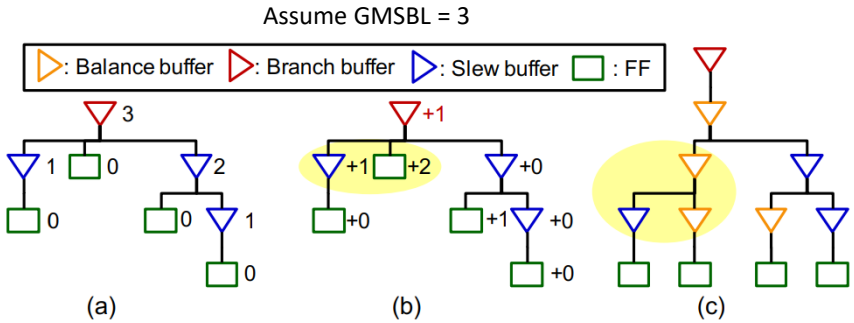
- Hold buffer: truck insertion
  - Special back to back flip-flop



approach	advantage	disadvantage
fix hold on data path	no setup penalty	large hold area cost
fix hold on clock branch	small hold area cost	setup penalty

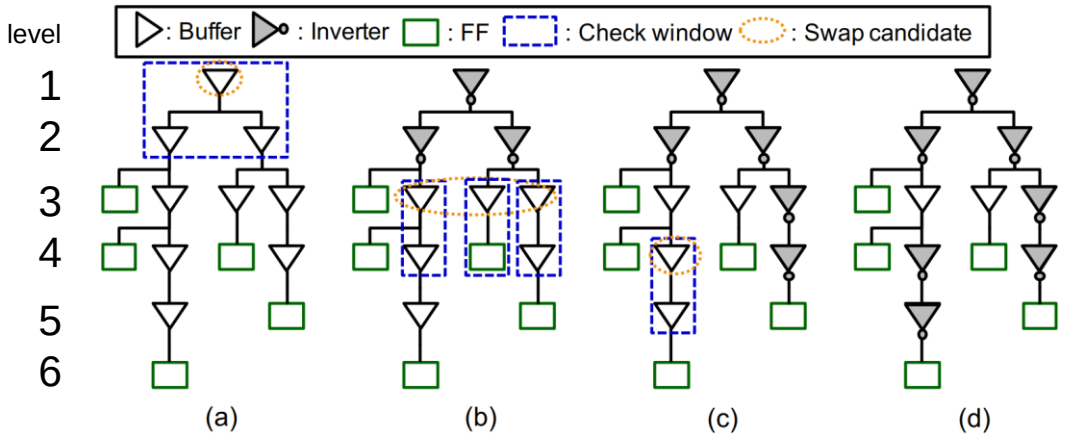
Local Tree Skew Balancing & OCV Minimization

- Insert balance buffer from branch buffer(local tree)
  - step1(a): bottom-up calculate current node level
  - step2(b): top-down propagate/push down balanced level to each node
    - GMSBL for balance level reference
  - step3(c): for each level insert required balancing buffer(s)
    - Note: merge/share level buffer concurrently

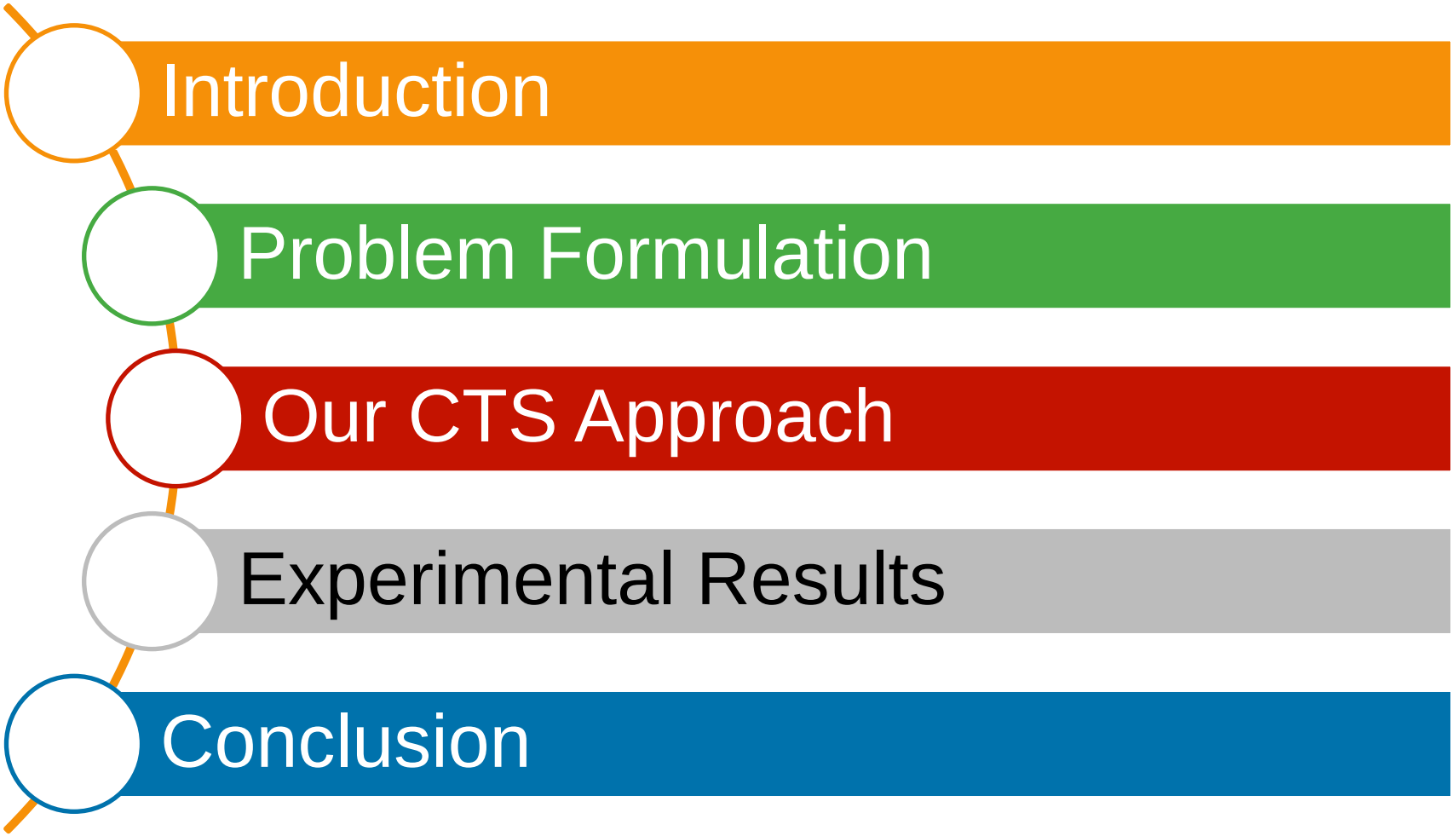


Duty Cycle Maintenance by Inverter Pair Swap

- Insert pair help to fix clock wave form – iterate level pair continuously
- Adjacent levels are all buffers ⇒ swap into inverters
  - (a)initial clock tree
  - (b)after first/second level inverter pair swap
  - (c)after third/fourth level inverter pair swap
  - (d)after fourth/fifth level inverter pair swap



Outline


Introduction
Problem Formulation
Our CTS Approach
Experimental Results
Conclusion

Experimental Setting

- Setting
  - Language: C++ with Tcl
  - Platform: Xeon Linux workstation with 2.1GHz CPU and 128GB memory
  - EDA tools: Synopsys ICC/ICC2
- Benchmark statistics
  - SHA256 bitcoin, 2019 tau contest circuit
  - TSMC 7/10nm process

ckt1(SHA256): 1core
ckt2(SHA256): 2core
ckt3(SHA256): 4core
ckt4(SHA256): 6core
ckt5(tau): vga_lcd
ckt6(tau): usb_funct

Circuit (Process)	Period (ns)	# of Instances	Area (um^2)	# of Nets	# of Flip-Flops	Pipe Depth
ckt1(T7)	1.53	183145	76053	262136	48016	127
ckt2(T7)	1.53	330929	138592	473348	115267	127
ckt3(T7)	1.53	654455	267800	908786	215112	127
ckt4(T7)	1.55	1164226	425849	1512791	314955	127
ckt5(T10)	1.42	69172	31132	55887	17079	6
ckt6(T10)	1.33	9836	3523	9990	1739	5

Comparison

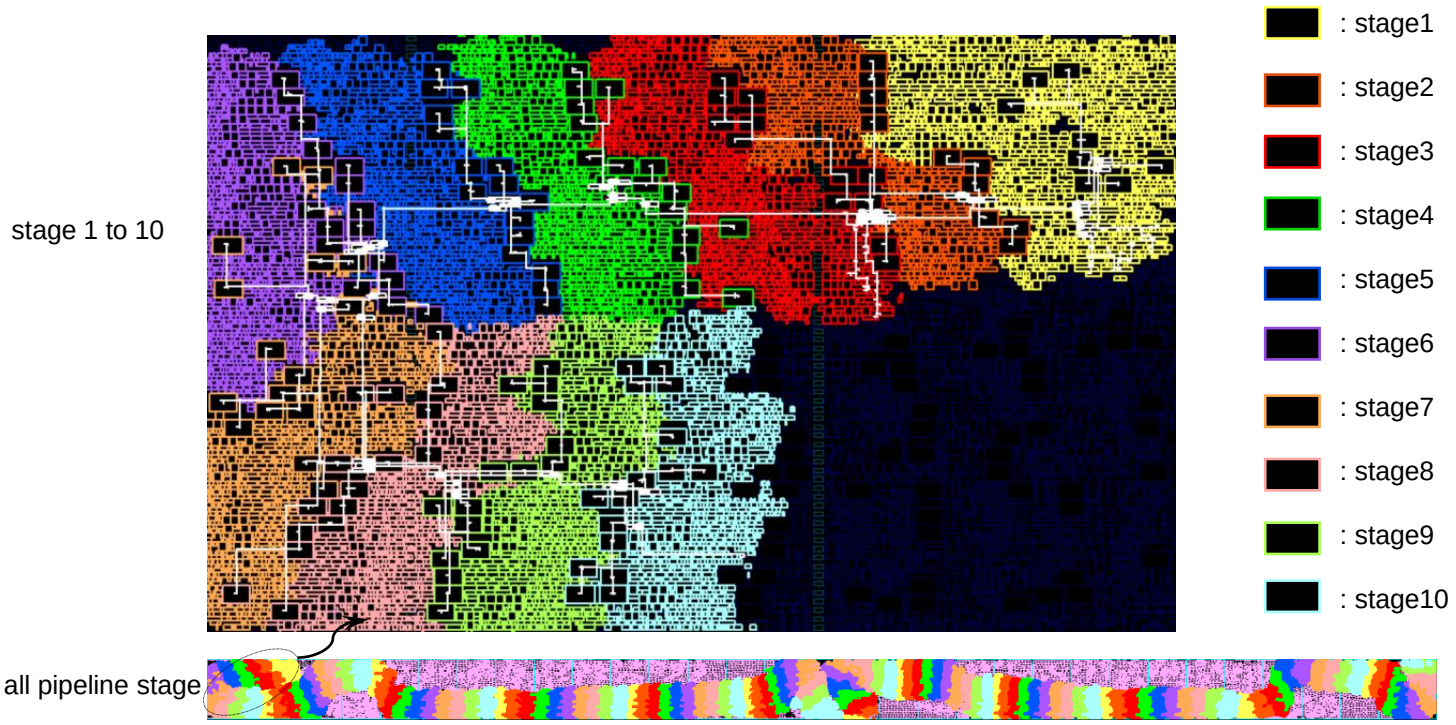
- VAT: Variation-Aware clock Tree
- MESH: clock hybrid tree-MESH based approach

Area Increase: CTS Repeater Area + Hold Area

Circuit	Method	No. of CTS Repeater	Ratio: CTS Repeater (buf/inv)	CTS Repeater Area(um^2)	Clock Power (W)	Clock Power Ratio	Max Latency (ns)	Global Skew (ns)	Setup WNS (ns)	Setup TNS (ns)	No. of Setup Vio.	Hold WNS (ns)	Hold TNS (ns)	No. of Hold Vio.	Hold Area (um^2)	Area Increase (um^2)	Area Increase Ratio
ckt1	VAT	2777	1034/1743	2223	5.88e-04	1	2.00	0.08	-0.0399	-0.6840	155	-0.1151	-280.3914	13212	4624	6847	1
	MESH	3327	3327/0	3684	8.32e-04	1.41	1.73	0.04	-0.0200	-0.3397	90	-0.1665	-178.2141	9689	3584	7268	1.06
	Ours	950	293/657	867	3.70e-04	0.62	5.69	5.43	-0.0170	-0.4377	108	-0.0596	-1.4891	183	74	941	0.13
ckt2	VAT	4721	1630/3091	3098	9.51e-04	1	1.80	0.12	-0.0681	-1.2933	191	-0.2325	-631.5519	24315	8267	11365	1
	MESH	4347	4347/0	4786	1.31e-03	1.38	2.94	0.11	-0.0288	-0.2907	61	-0.1322	-66.6633	5486	2029	6815	0.59
	Ours	1686	512/1174	1508	6.50e-04	0.68	5.29	4.97	-0.0465	-2.9272	493	-0.0354	-1.3256	134	54	562	0.04
ckt3	VAT	8608	3020/5588	5560	4.63e-03	1	2.02	0.12	-0.0652	-5.6820	881	-0.1468	-1229.1501	46807	15446	21006	1
	MESH	6390	6390/0	6691	6.63e-03	1.43	1.94	0.11	-0.0569	-1.2742	337	-0.1133	-68.6439	6896	2551	9242	0.43
	Ours	3604	838/2766	3210	4.16e-03	0.90	6.91	6.62	-0.0980	-11.2637	1571	-0.0508	-3.2947	270	110	3320	0.15
ckt4	VAT	14706	4766/9940	10508	2.43e-02	1	2.07	0.14	-0.0693	-4.9077	681	-0.1950	-2272.9488	75408	24884	35392	1
	MESH	9382	9382/0	10374	3.41e-02	1.40	2.08	0.08	-0.0259	-0.8387	156	-0.1530	-196.5100	14291	5001	15375	0.43
	Ours	5369	1238/4131	4787	2.20e-02	0.90	11.77	11.41	-0.0912	-18.9870	1962	-0.3029	-1.9930	129	56	4843	0.13
ckt5	VAT	4246	1790/2456	859	2.73e-04	1	0.51	0.12	0.0000	0.0000	0	-0.0600	-0.1600	231	592	1451	1
	MESH	1410	1410/0	1559	7.03e-04	2.57	0.54	0.04	0.0000	0.0000	0	0.0100	0.0000	0	0	1559	1.07
	Ours	959	419/521	1268	2.57e-04	0.94	2.53	2.30	0.0000	0.0000	0	0.0100	0.0000	0	0	1268	0.87
ckt6	VAT	451	221/230	204	6.99e-05	1	0.28	0.05	0.0200	0.0000	0	-0.0200	-0.0500	5	3	207	1
	MESH	256	256/0	283	7.46e-05	1.06	0.39	0.30	0.0200	0.0000	0	-0.0200	-0.0500	10	5	288	1.39
	Ours	154	52/102	141	6.12e-05	0.87	0.81	0.77	0.0200	0.0000	0	-0.0200	-0.0500	0	0	141	0.68
Ratio	VAT	1	35:65	1	1		1	1				1	1	1	1	1	
	MESH	0.71	100:0	1.22	1.41		1.11	1.08				0.77	0.12	0.23	0.24	0.53	
	Ours	0.36	26:74	0.52	0.88		3.80	50.00				0.61	0.0018	0.0089	0.0054	0.15	

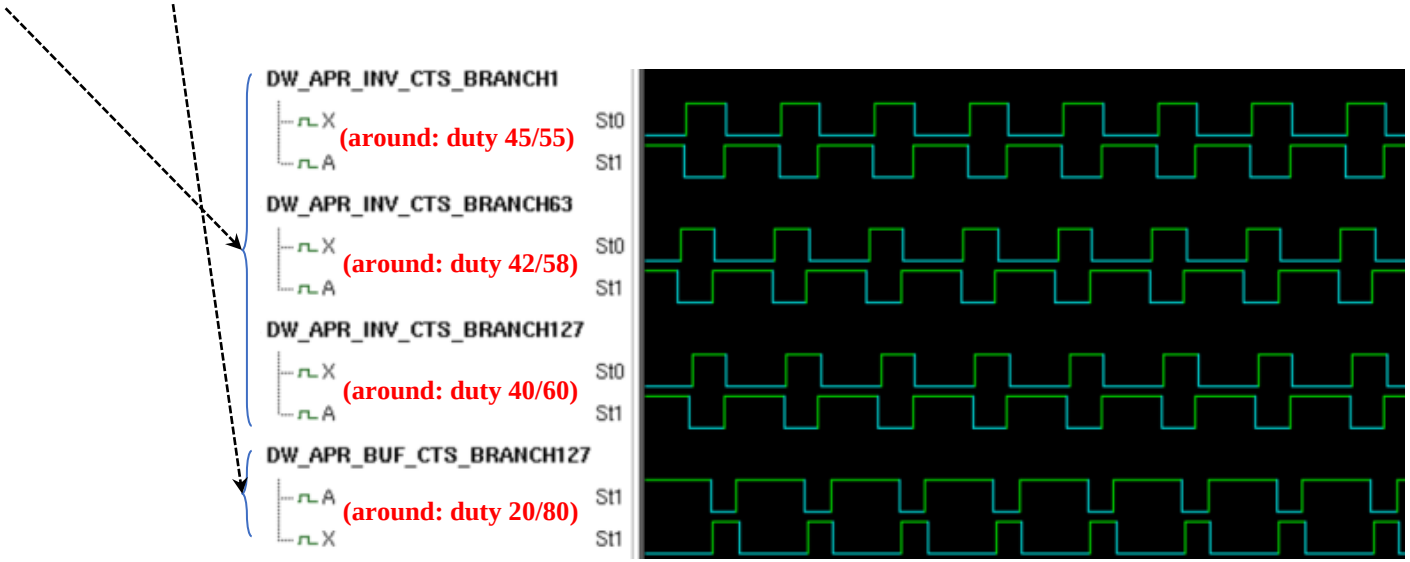
Pipe Related Placement Example

- ckt1 SHA256 one core
  - For timing convergent, pipe flip-flop relating placement

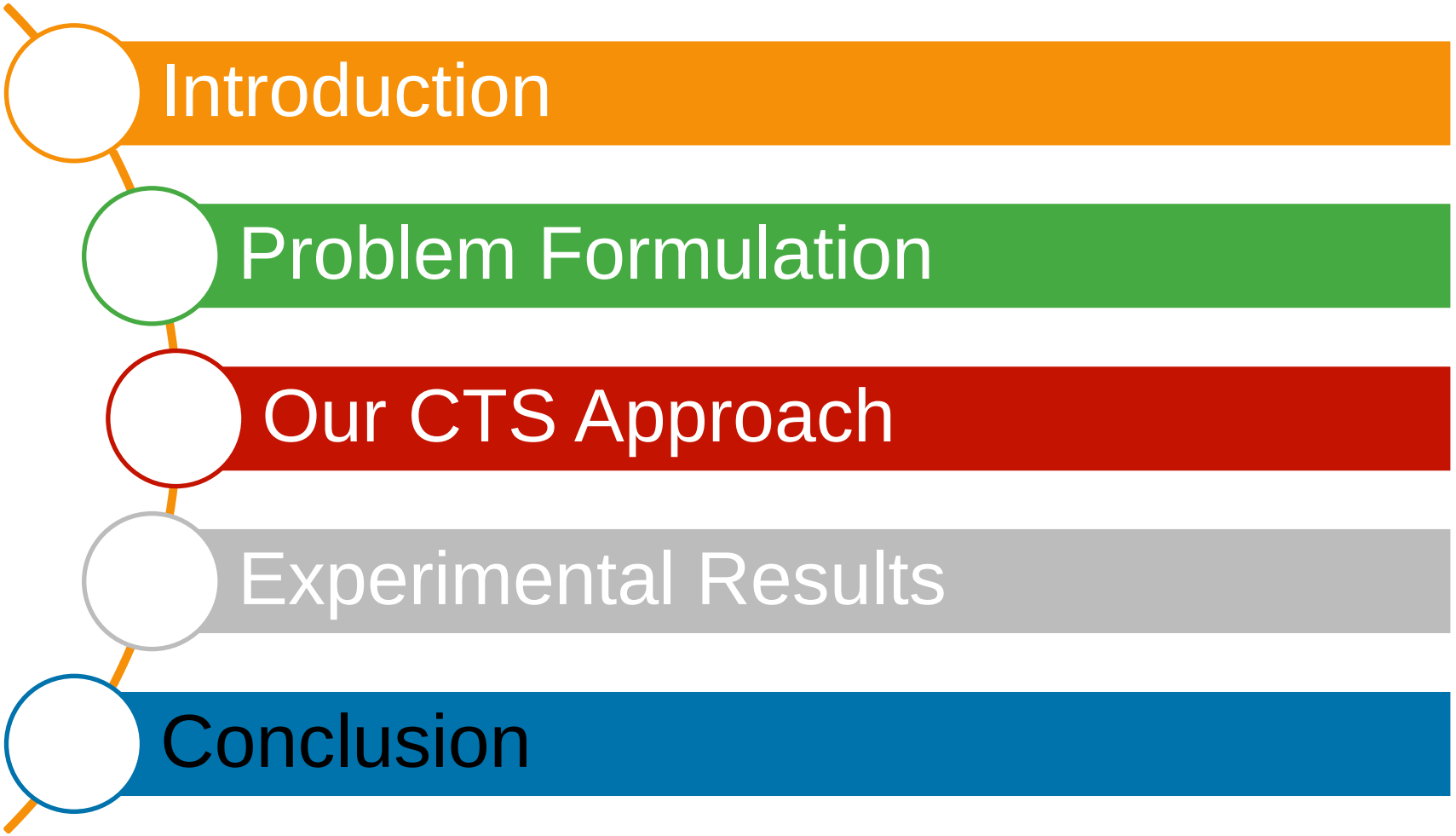


Duty Cycle Waveform

- ckt3 duty cycle waveform of branch
  - Inverter vs. buffer



Outline


Introduction
Problem Formulation
Our CTS Approach
Experimental Results
Conclusion

Conclusion

- Clock feeds from last pipeline stage backward to the first one
  - Clock latency difference between two consecutive stages ─ constant delay
  - Maximize common clock path – reduce OCV effect
  - Utilize setup time slack to gain hold time margin
  - Inverter pair swap improve duty cycle waveform
- Minimize repeater and hold delay cell area
- Applicable to deep pipeline and strong data flow

Thank You!