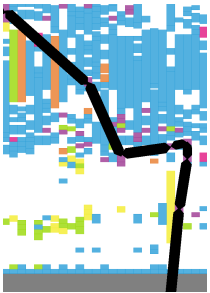
Delay Optimization of Combinational Logic by AND-OR Path Restructuring

Ulrich Brenner, Anna Silvanus

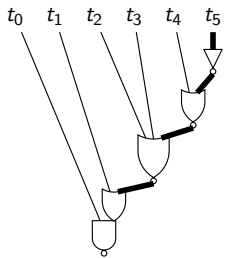
January 19, 2022
27th Asia and South Pacific Design Automation Conference

AND-OR Paths as Delay-Critical Paths on Computer Chips

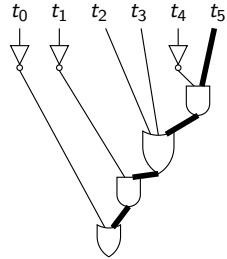
A combinational path on a computer chip can be translated into an AND-OR path and optimized as such.



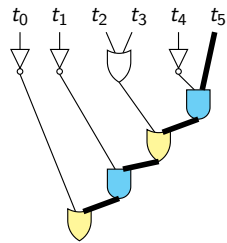
Part of a delay-critical path on a chip.



Gates of delay-critical path.



Inversions pushed aside via De Morgan.



Extracted AND-OR path with side gates.

AND-OR Path Optimization

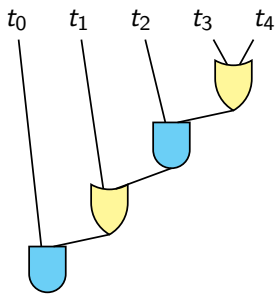
Definition (And-Or Path)

An AND-OR path on inputs $t_0, \ldots, t_{m-1}$ is a Boolean formula of type

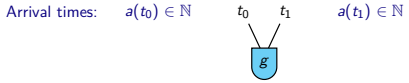
$$g(t_0, \ldots, t_{m-1}) = t_0 \wedge (t_1 \vee (t_2 \wedge (t_3 \vee (t_4 \wedge (\ldots t_{m-1}) \ldots)) \text{ or}$$

$$g^*(t_0, \ldots, t_{m-1}) = t_0 \vee (t_1 \wedge (t_2 \vee (t_3 \wedge (t_4 \vee (\ldots t_{m-1}) \ldots)).$$

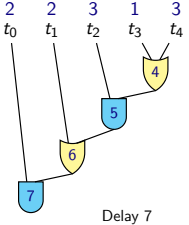
Example: $g(t_0 \wedge (t_1 \vee (t_2 \wedge (t_3 \vee t_4))))$:



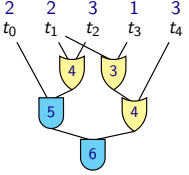
AND-OR Path Optimization Problem



The delay of a circuit C on inputs $t_0, \ldots, t_{m-1}$ with arrival times $a(t_i) \in \mathbb{N}$ is the maximum arrival time of any node in C.



Delay 7



Delay 6

Task: Given inputs $t = (t_0, \ldots, t_{m-1})$ and input arrival times a, find a circuit with minimum delay realizing an AND-OR path on t using only AND2 and OR2 gates.

Our Contributions

And-Or path optimization:

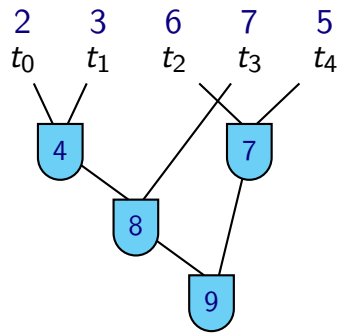
- Improved dynamic programming algorithm
- In experiments: significantly better than previous approaches and in most cases optimum

Application in timing:

- New logic restructuring framework
- Still improves timing after classical timing-optimization (gate sizing, buffering etc.) on recent industrial logic chips

Much Easier Case: Optimization of AND-Trees

Greedy algorithm (Huffman Coding) finds optimum solution:



Huffman Coding for Symmetric Trees

For given inputs $t = (t_0, \ldots, t_{m-1})$ with arrival times $a(t_i)$, let

$$W(t) := \sum_i 2^{a(t_i)}.$$

Theorem (Kraft; Huffman; Golumbic; Van Leeuwen)

For inputs $t = (t_0, \ldots, t_{m-1})$ with arrival times $a(t_i)$, the Huffman Coding algorithm constructs a binary And tree (or Or tree) on t with delay exactly $\lceil \log_2 W(t) \rceil$. The algorithm can be implemented in linear time after sorting.

Observation

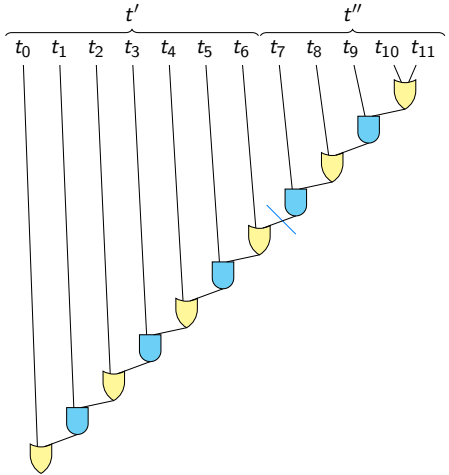
Any circuit containing only 2-input gates on inputs $t_0, \ldots, t_{m-1}$ with arrival times $a(t_i)$ has delay at least $\lceil \log_2 W(t) \rceil$.

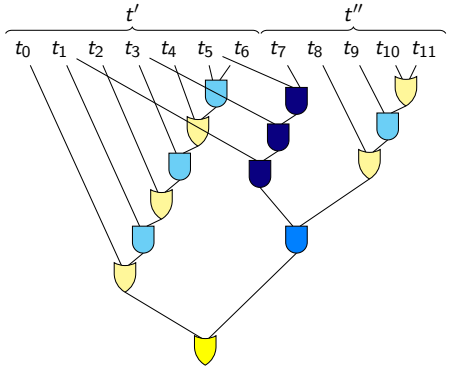
$\Rightarrow \lceil \log_2 W(t) \rceil$ is also a lower bound on the delay of any AND-OR path circuit.

Well-Known Recursive Circuit Construction

Idea

Recursively split the AND-OR path into smaller AND-OR paths plus additional logic.



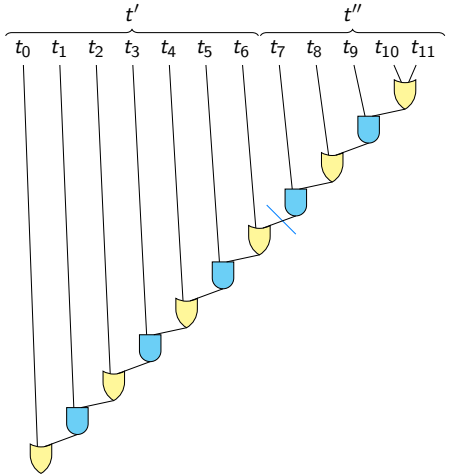


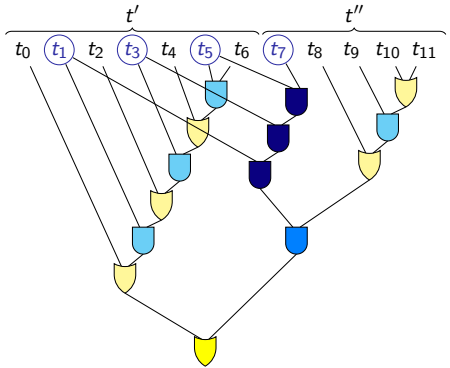
Equivalent circuit obtained by a single split.

Well-Known Recursive Circuit Construction

Idea

Recursively split the AND-OR path into smaller AND-OR paths plus additional logic.





Equivalent circuit obtained by a single split.

Well-Known Recursive Circuit Construction

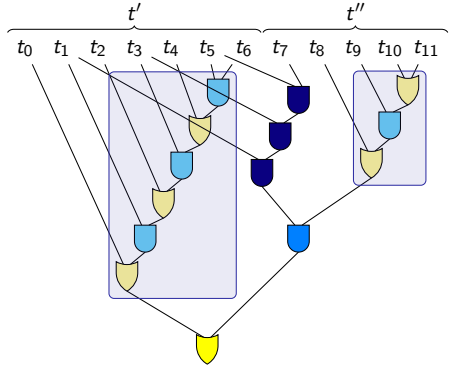
Idea

Recursively split the AND-OR path into smaller AND-OR paths plus additional logic.

Circuit quality depends on used splits.

Classical approaches: Prefix circuits, e.g., [Sklansky, 60] [Kogge,Stone, 73], [Ladner, Fischer, 80], [Brent,Kung82], [Rautenbach, Szegedy, Werber, 06], [Held, Spirkl, 17b]

More general splits: e.g., [Khrapchenko, 67], [Rautenbach, Szegedy, Werber, 03], [Grinchuk, 08], [Spirkl, 14], [Held, Spirkl, 17b]



Prefix-split recursion.

Well-Known Recursive Circuit Construction

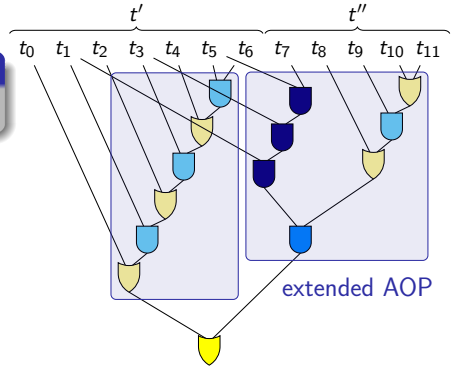
Idea

Recursively split the AND-OR path into smaller AND-OR paths plus additional logic.

Grinchuk's approach

Optimize extended AND-OR paths.

$\Longrightarrow$ [Grinchuk, 08], [Commentz-Walter, 79]: Optimum depth up to an additive constant.



[Grinchuk, 08] recursion.

Extended AND-OR Paths

Definition

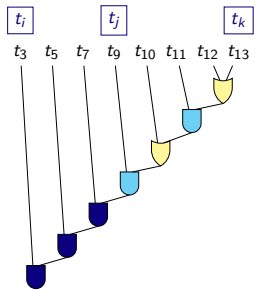
Given inputs $t = (t_0, \ldots, t_{m-1})$, an extended AND-OR path is a function of type

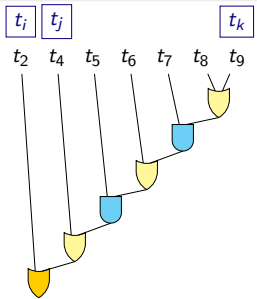
$$\phi_{i,j,k} = t_i \wedge t_{i+2} \wedge \ldots \wedge t_{j-4} \wedge t_{j-2} \wedge g(t_j, \ldots, t_k)$$

or

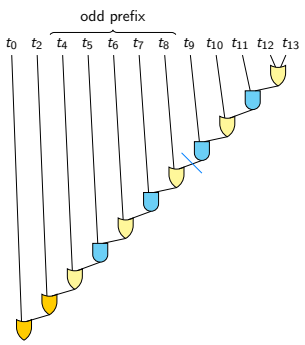
$$\phi^*_{i,j,k} = t_i \vee t_{i+2} \vee \ldots \vee t_{j-4} \vee t_{j-2} \vee g^*(t_j, \ldots, t_k)$$

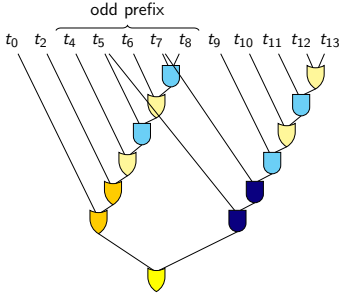
with $0 \leq i \leq j \leq k < m$ and $j - i$ even.





Alternating Split with an Odd Prefix



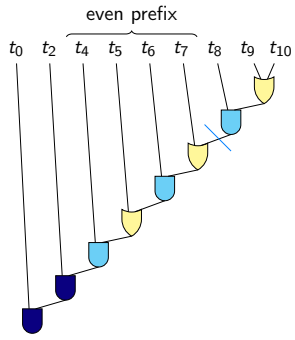


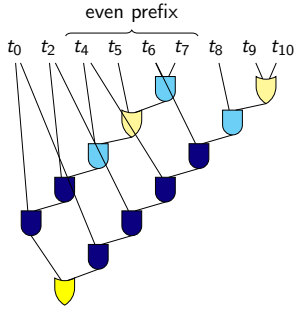
Alternating Split with an Odd Prefix

For odd prefix length $2\lambda + 1$ with $\lambda \in \{0, \ldots, \frac{k-j-1}{2}\}$, we have

$$\phi_{i,j,k} = \phi_{i,j,j+2\lambda} \wedge \phi^*_{j+1,j+2\lambda+1,k}, \quad \text{and}$$
$$\phi^*_{i,j,k} = \phi^*_{i,j,j+2\lambda} \vee \phi_{j+1,j+2\lambda+1,k}.$$

Alternating Split with an Even Prefix





Alternating Split with an Even Prefix

For even prefix length $2\lambda + 1$ with $\lambda \in \{0, \ldots, \frac{k-j-1}{2}\}$, we have

$$\phi_{i,j,k} = \phi_{i,j,j+2\lambda-1} \vee \phi_{i,j+2\lambda,k} \quad \text{and}$$
$$\phi^*_{i,j,k} = \phi^*_{i,j,j+2\lambda-1} \wedge \phi^*_{i,j+2\lambda,k}\,.$$

Split Options in Our Dynamic Program

Possible Splits

For odd prefix length $2\lambda + 1$ with $\lambda \in \{0, \ldots, \frac{k-j-1}{2}\}$, we have

$$\phi_{i,j,k} = \phi_{i,j,j+2\lambda} \wedge \phi^*_{j+1,j+2\lambda+1,k}\,, \tag{1}$$

for even prefix length 2λ with $\lambda \in \{1, \ldots, \frac{k-j}{2}\}$, we have

$$\phi_{i,j,k} = \phi_{i,j,j+2\lambda-1} \vee \phi_{i,j+2\lambda,k}\,, \tag{2}$$

and we have

$$\phi_{i,j,k} = \phi_{i,j-2,j-2} \wedge \phi_{j,j,k}\,. \tag{3}$$

Previous Work [B., Hermann, 2019]

Delay bound of $\log_2 W + \log_2 \log_2 m + \log_2 \log_2 \log_2 m + 4.3$.

Simple Dynamic Program for AND-OR Path Optimization

Input: Inputs $t = (t_0, \ldots, t_{m-1})$ with arrival times $a(t_i) \in \mathbb{N}$.
Output: A Boolean circuit computing $f(t)$.

for $l \leftarrow 1$ **to** m **do**
  foreach $0 \leq i \leq j \leq k < m$ with $j - i$ even s.t. $\phi_{i,j,k}$ has l inputs **do**
    if $k \in \{j, j+1\}$ **then**
      Apply Huffman coding to construct an optimum circuit $C_{i,j,k}$ for $\phi_{i,j,k}$ and an optimum circuit $C^*_{i,j,k}$ for $\phi^*_{i,j,k}$.
    else
      $\mathcal{C}$:= list of candidate circuits for $\phi_{i,j,k}$ arising from applying any valid split (1), (2), (3).
      $C_{i,j,k}$:= delay-minimum circuit among $\mathcal{C}$.
      $C^*_{i,j,k}$:= dual circuit of $C_{i,j,k}$.

return $C_{0,0,m-1}$

Simple Dynamic Program – Guarantees

- All mentioned approaches are generalized by our algorithm.
- All other mentioned theoretical guarantees also hold.

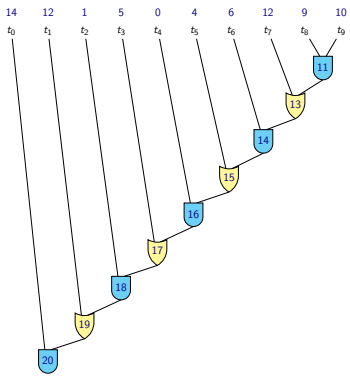
Theorem

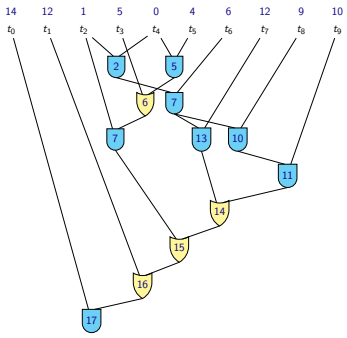
Given Boolean input variables $t = (t_0, \ldots, t_{m-1})$ *with arrival times* $a\colon \{t_0, \ldots, t_{m-1}\} \to \mathbb{N}$*, the dynamic program computes a circuit* C *realizing* $f(t)$ *with delay at most*

$$\mathit{delay}(C) \leq \log_2 W + \log_2 \log_2 m + \log_2 \log_2 \log_2 m + 4.3$$

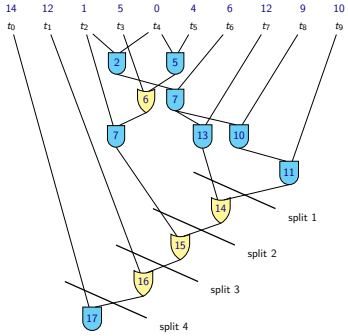
and can be implemented to run in time $\mathcal{O}(m^4)$*.*

Example Solution of Simple DP

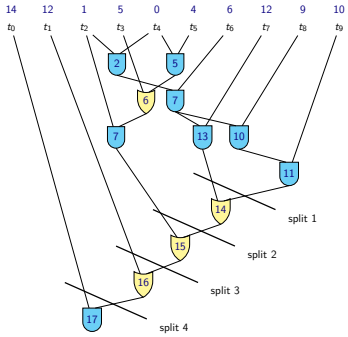


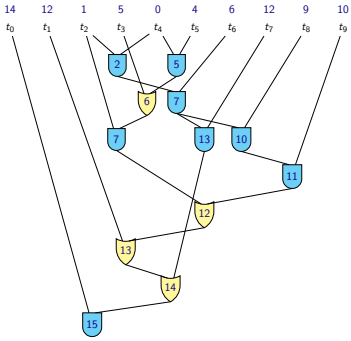


Problem with Simple DP



Problem with Simple DP





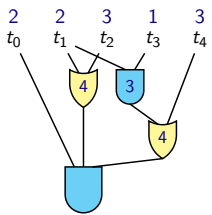
Undetermined circuits

Definition

An undetermined circuit is a Boolean circuit C consisting of AND and OR gates only such that all gates with the possible exception of out have fan-in two. With given input arrival times, the weight of C is

$$\text{weight}(C) := \sum_{i=1}^{k} 2^{d_i},$$

here $d_1, \ldots, d_k$ are the arrival times at the predecessors of out.



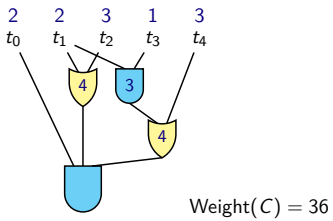
An undetermined circuit with weight $2^2 + 2^4 + 2^4 = 36$.

Lemma

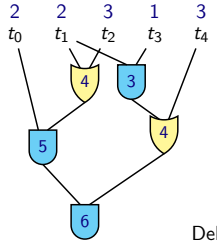
Given an undetermined circuit C, we can construct a Boolean circuit using AND2 and OR2 gates only that computes the same Boolean function as C with delay at most $\lceil \log_2(\text{weight}(C)) \rceil$.

Proof.

Apply Huffman coding with the predecessors of out as inputs. □

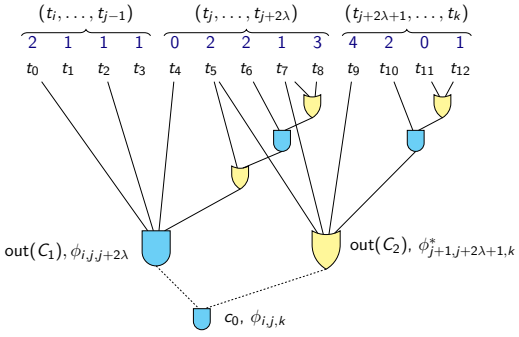


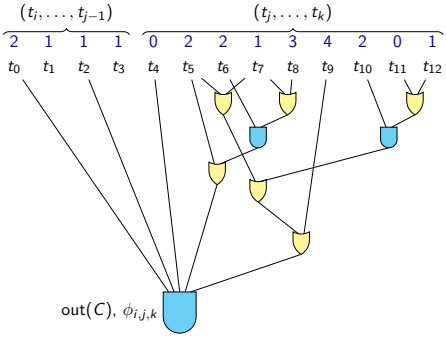
Weight$(C) = 36$



Delay $= 6$

Merging Undetermined Circuits





Final Dynamic Program for AND-OR Path Optimization

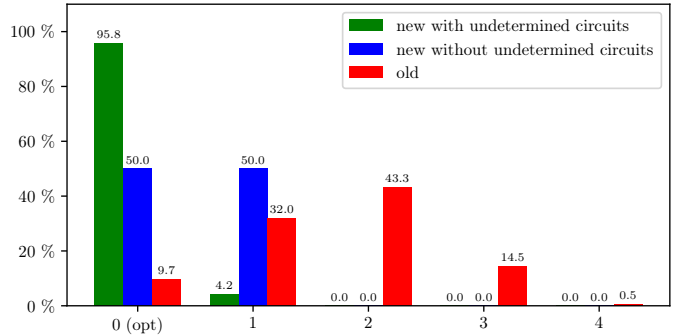
Input: Inputs $t = (t_0, \ldots, t_{m-1})$ with arrival times $a(t_i) \in \mathbb{N}$.
Output: A Boolean circuit computing $f(t)$.

for $l \leftarrow 1$ **to** m **do**
  foreach $0 \le i \le j \le k < m$ with $j - i$ even even s.t. $\phi_{i,j,k}$ has l inputs **do**
    if $k \in \{j, j+1\}$ **then**
      Apply Huffman coding to construct an optimum circuit $A_{i,j,j}$ for $\phi_{i,j,j}$ and an optimum circuit $O_{i,j,j}$ for $\phi^*_{i,j,j}$.
    else
      $\mathcal{C}$:= list of candidate undetermined circuits for $\phi_{i,j,k}$ arising from applying any valid split (1), (2) followed by a merge operation.
      $A_{i,j,k}$:= weight-minimum circuit among $\mathcal{C}$ with output gate AND.
      $O_{i,j,k}$:= weight-minimum circuit among $\mathcal{C}$ with output gate OR.

C := weight-minimum undetermined circuit among $A_{0,0,m-1}$ and $O_{0,0,m-1}$.
Transform C into a circuit C' over $\{\text{AND2}, \text{OR2}\}$.
return C'

Comparison on Instances of Artificial Testbed

- For each $n \in [4, 28]$, create 1000 instances with random arrival times in the interval $[0, n]$.
- Compute delay difference to optimum solution

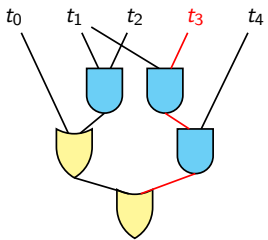


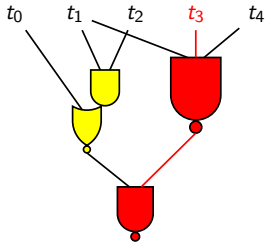
Old: Previous algorithm used in practice ([Rautenbach, et al., 2006], [Held, Spirkl, 2017])

Technology Mapping

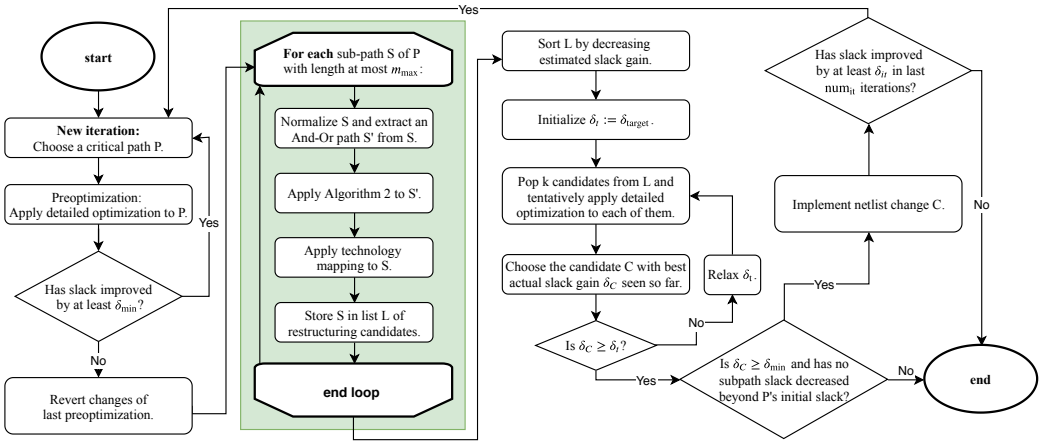
Idea

- Virtual timing model
- Objectives: slack, area and netlength
- Dynamic program: Merge gates locally and apply De Morgan's laws.
- On instances with few cycles: FPTAS
  - Applied after path restructuring and SymTree optimization





Our logic optimization framework



Experiments with the whole framework

Instances

- Industrial 7nm logic chips from IBM
- Between 22k and 332k gates.
- All instances result from a timing-driven placement followed by timing-optimization steps including gate sizing and buffering.
- Classical timing optimization cannot improve these instances any more.

We show the effect of the overall framework.

Results on 7nm Real-World Instances

Unit	Run	WS [ps]	TNS [ns]	# Gates	Area	Netlength	WACE5	Time [s]
i1	init	−107	−26.1	22 412			83 %	
	LO	−104	−26.0	22 431	+0.01 %	0.00 %	82 %	409
i2	init	−14	−1.7	38 048			93 %	
	LO	−14	−1.6	38 067	+0.02 %	0.00 %	93 %	50
i3	init	−65	−67.4	64 230			97 %	
	LO	−53	−57.2	64 249	+0.04 %	+0.09 %	96 %	140
i4	init	−17	−1.1	78 193			110 %	
	LO	−3	−0.1	77 851	−0.28 %	−0.14 %	110 %	230
i5	init	−174	−335.4	212 210			94 %	
	LO	−152	−332.9	212 236	+0.01 %	+0.01 %	94 %	306
i6	init	−39	−19.7	268 473			87 %	
	LO	−24	−13.6	268 336	0.00 %	+0.03 %	88 %	272
i7	init	−69	−182.8	274 723			95 %	
	LO	−55	−168.9	274 863	+0.03 %	+0.02 %	95 %	400
i8	init	−125	−656.3	332 695			92 %	
	LO	−116	−640.9	332 787	0.00 %	+0.02 %	92 %	253

Thank you for listening!