



Timing Optimization Techniques for FPGAs



Outline

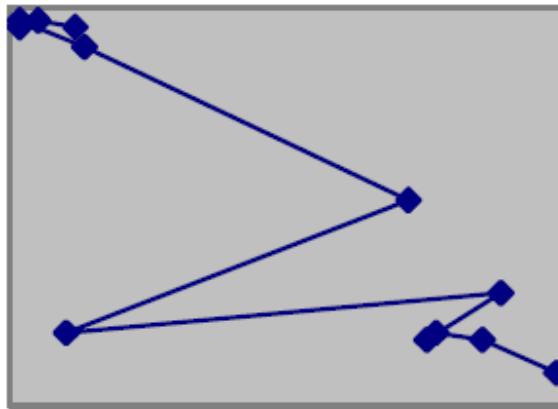
- Introduction
- Post-Placement Timing Optimization by Logic Replication
- Post-Routing Timing Optimization by LUT Output Polarity Selection

Introduction

- Due to some special architectural characteristics of FPGA, there are some unique opportunities for timing optimization not found in ASIC.
- We consider examples of post-placement and post-routing timing optimization.

Circuit Delay and Mon-Monotone Paths

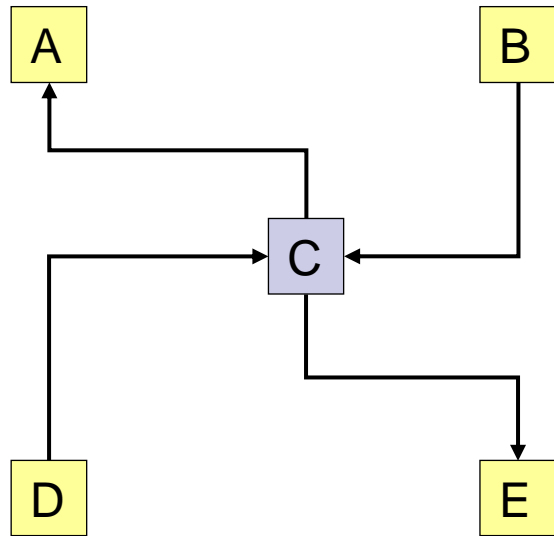
- Clock period depends on critical path delay which is dominated by interconnect delay
- Even with timing-driven placement, typical critical paths are non-monotone



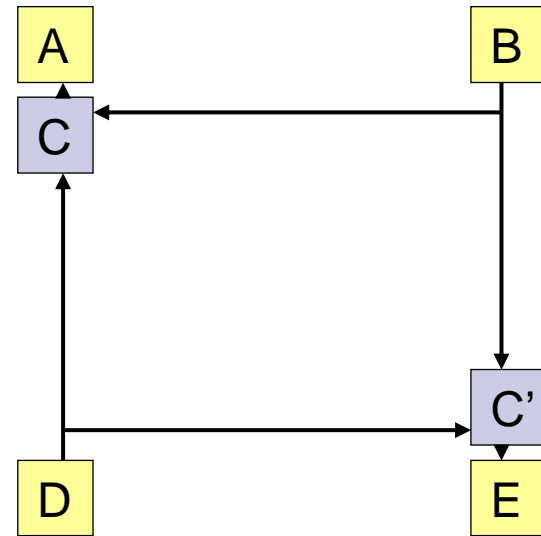
- “Straightening” a non-monotone path can reduce its delay, but ...

Timing Optimization by Logic Replication

- Moving one cell to straighten one non-monotone path may degrade other paths
- Solution: Replicate a cell



Original placement



With replication and relocation

- We can take advantage of the unused LEs in a FPGA

Duplication Algorithm

Input: A placed circuit in a FPGA

Output: A modified placement with cell replication

```
1. Duplication() {  
2.     Timing Analysis  
3.     While (!done) {  
4.         Candidate selection  
5.         Destination selection  
6.         Node duplication  
7.         Fanout partitioning  
8.         Incremental legalization  
9.         Undo when timing gets worse  
10.    }  
11. }
```

Duplication Algorithm

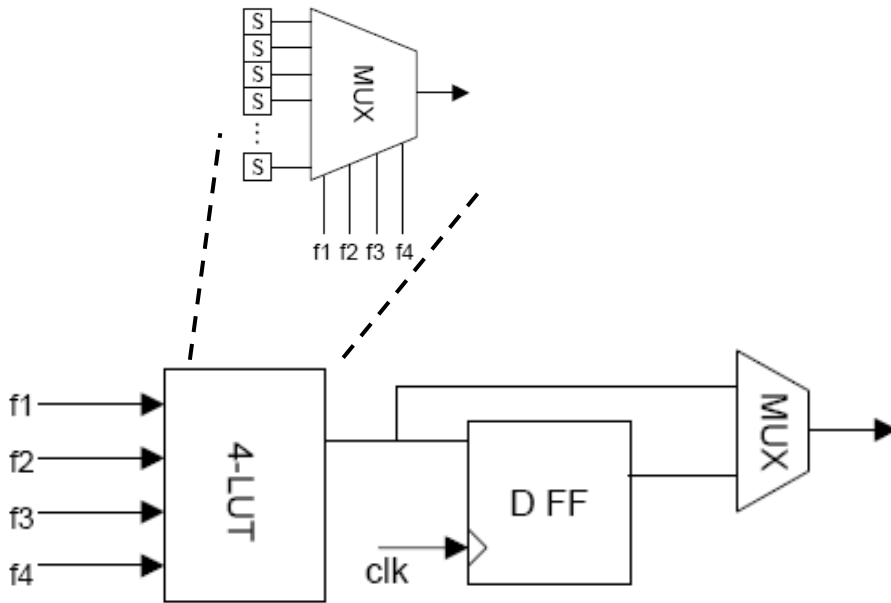
- Use local monotonicity to guide replication
 - Select critical path cell with high deviation from its local monotone region:
$$deviation(i) = dist(prev(i), i) + dist(i, next(i)) - dist(prev(i), next(i))$$
 - Place cell again within its local monotone region
- Careful fanout partitioning to avoid degrading other paths
- Legalization removes cell overlap introduced by replication

Experimental Results

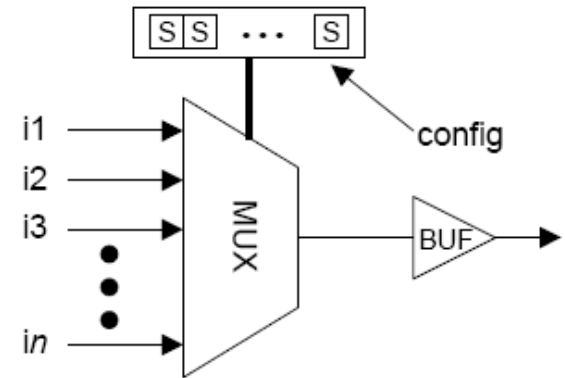
- Apply logic replication based timing optimization after placement
 - Only modest increase ($< 1\%$) in #LEs used
 - Reduce worst path delay by 13% on average
 - Increase total wirelength by $\sim 3\%$.

Delay Characteristics in FPGA

- Significant difference in rise and fall delays in logic blocks and buffered routing MUXes
 - due to mobility difference in P and N transistors



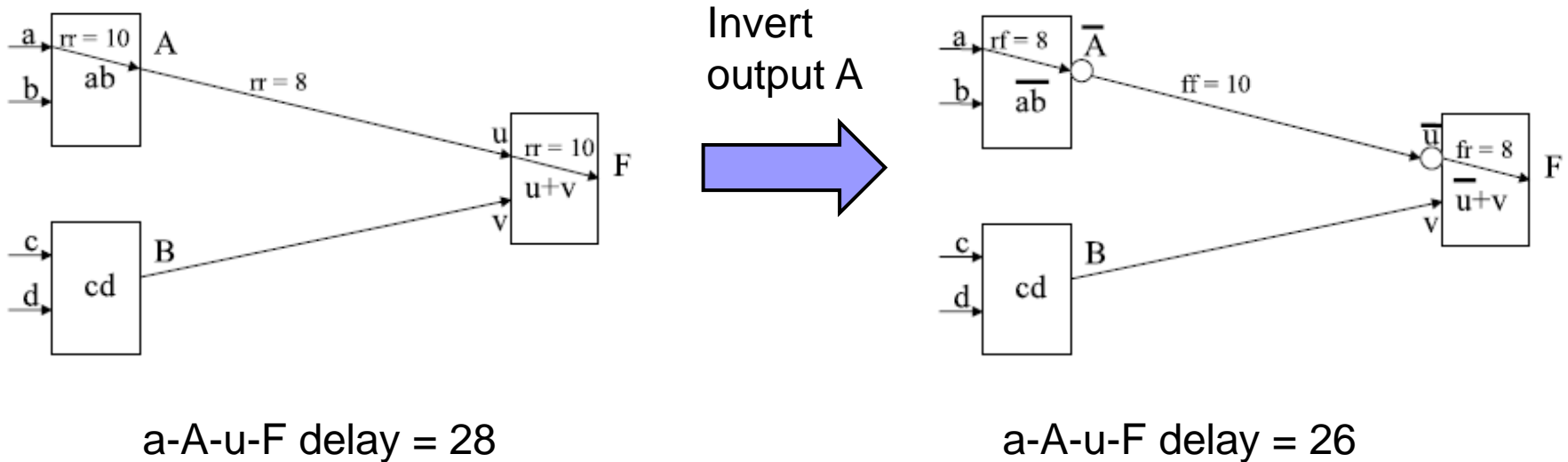
Logic block



Routing switch

Timing Optimization by LUT Output Polarity Selection

- LUT output polarities impact path delays



- Timing optimization by output polarity selection after P&R has 0 cost.

Polarity Selection Algorithm

■ Assumptions

- Circuit already placed & routed
- A timing graph is constructed where each pin is modeled by a node

■ Algorithm

- *Phase 1: Compute delay tuples*
 - Compute possible rise and fall delays for each node, that reflect different polarity settings
 - Process nodes in topological order (from PIs to POs)
- *Phase 2: Output polarity selection*
 - Select polarity of each LUT output to optimize rise and fall delays at POs

Phase 1: Compute Delay Tuple Sets

Input: Timing graph $G = (V, E)$

Output: Minimum delay tuple sets of nodes in G

1. Sort nodes in $G = (V, E)$ in topological order;
2. Initialize all PIs with a delay tuple $(0, 0, -)$;
3. for every node $u \in V$ in topological order {
4. Compute delay tuples of u if polarity of u is unchanged;
5. Compute delay tuples of u if polarity of u is changed;
6. Prune dominated delay tuples of u ;
7. }

Rise and fall delays, and polarity of a node u
are stored as a tuple $(d^r(u), d^f(u), pol(u))$

Phase 1: Compute Delay Tuple Sets

- # tuples formed at node u
 - $n_u = 2 \prod_{1 \leq i \leq k} n_i$
where $n_i = \#$ tuples in i -th fanin of u
 - Grow exponentially!
- Prune dominated tuples
 - Tuple $t1 = (d^r_1(u), d^f_1(u), pol_1(u))$ dominates $t2 = (d^r_2(u), d^f_2(u), pol_2(u))$ if
 - $t1$ and $t2$ have same polarity and both rise and fall delays of $t1$ are less than or equal to those of $t2$, i.e.,
 $pol_1(u) = pol_2(u)$, $d^r_1(u) \leq d^r_2(u)$ and $d^f_1(u) \leq d^f_2(u)$

Phase 2: Polarity Selection

- If timing graph is a tree, we can optimally backtrack the optimal polarities for all LUT outputs in reverse topological order.
- Otherwise, output polarity selection to optimize two different paths may conflict, so we can only select output polarities heuristically to reduce circuit delay.

Experimental Results

- Apply polarity selection based timing optimization after place & route
 - Delay reduced by 2.5% on average

References

- “Timing Optimization of FPGA Placements by Logic Replication”, in *DAC'03*
- “Post-Route LUT Output Polarity Selection for Timing Optimization”, in *FPGA'07*