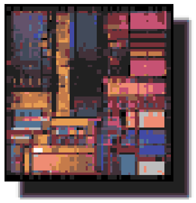
ECE 595Z
Digital VLSI Design Automation

Module 6 (Lectures 21-24): Timing Analysis and Optimization

Lecture 21



Anand Raghunathan

MSEE 318

raghunathan@purdue.edu

Technology mapping: Re-cap

- Technology mapping is a DAG covering problem
  - Convert circuit to be mapped into subject graph, cells from library into pattern graphs
- Two solution approaches discussed
  - Binate covering formulation – considers arbitrary DAGs, but not very scalable in practice
  - Tree covering using dynamic programming – divide DAG into trees, utilize optimal and efficient algorithm to map each tree
    - Inverter heuristic creates additional opportunities for matches, improves solution quality

Technology mapping: Remaining questions

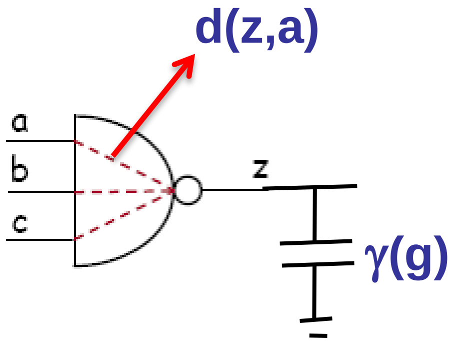
- How to map for minimum delay?
  - Not an additive cost function
  - Delay of a cell depends on its drivers and loads
- How to partition a DAG into trees

Technology Mapping for Delay

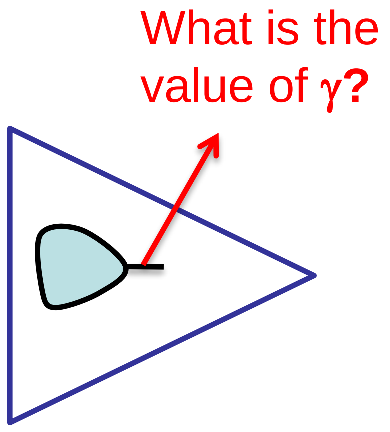
- Delay of a gate

$$d(g,i) = \alpha(g,i) + \beta(g).\gamma(g)$$

  - **α(g,i)**: intrinsic delay of ***g*** from input ***i***
  - **β(g)**: delay per unit load
  - **γ(g)**: capacitive load being driven



- Problem: In tree mapping, γ is NOT known when finding the best match at a vertex
  - γ is determined by matches at fanout



Technology Mapping for Delay : Constant Delay Model

- **Simplification**: Constant delay model
  - Delay of gate is independent of load it drives
- In this case, dynamic programming still works!

```
int min_delay_const_load(v, P){
/* v is a vertex in the tree, P is the set of pattern graphs */
        best_cost = infinity;
        foreach(m = match(v, P)) {
                cost(m) = $\max_{v_i \in inputs(m)}$ ( $\alpha(m,i) + \beta(m)\,\gamma_0$ +
                                min_delay_constant_load($v_i$, P) );
                if(cost(m) < best_cost){
                        match(v) = m;
                        best_cost = cost(m);
                }
        }
        return best_cost;
}
```

Technology Mapping for Delay : Load-dependent Delay Model

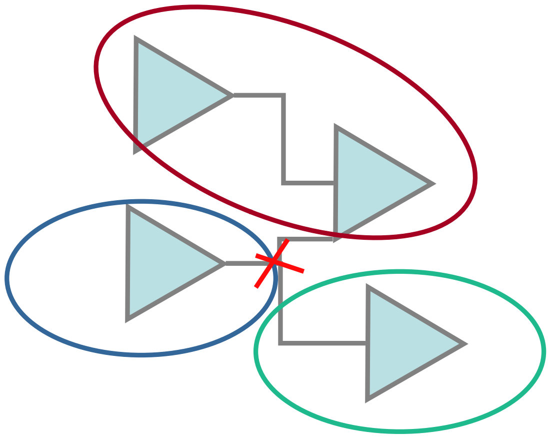
- Consider a set of discrete loads $\Gamma(v) = \{\gamma_1, \gamma_2, \ldots \gamma_k\}$ for vertex v.
  - can be obtained by dividing useful range into bins
- Find best solution for each load at each vertex

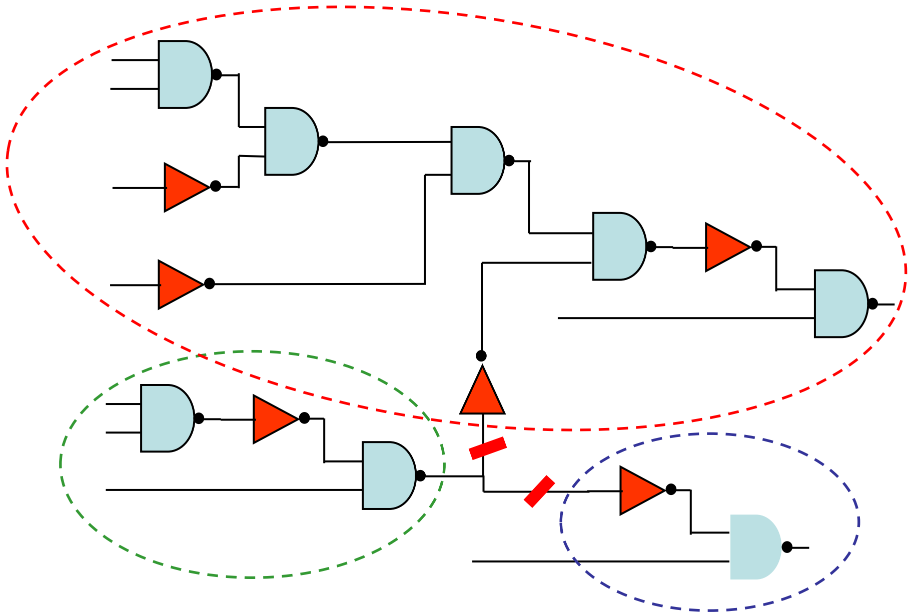
```
int min_delay(v, Γ(v), P){

        foreach(γ ∈ Γ(v)) {
                best_cost(v, γ) = infinity;
        }

        foreach(m = match(v, P), γ ε Γ(v)){
                cost(m, γ) = max_{v_i ∈inputs(m)} ( α(m,i) + β(m) γ +
                                                  min_delay(v_i, γ(m,i), P) );
                /* γ(m,i) is input capacitance of match m at input i */
                if(cost(m, γ) < best_cost(v, γ)){
                        match(v, γ) = m;
                        best_cost(v, γ) = cost(m, γ);
                }
        }
        return best_cost(v, γ_0);
}
```

- Final step: Given a load at the root of the tree, a backward traversal from the root to the leaves is needed at the end to pick the appropriate matches.

Partitioning a DAG into Trees

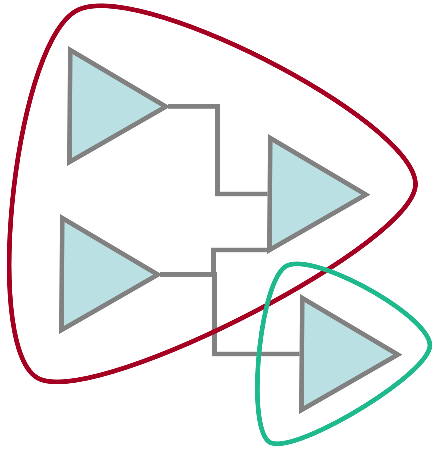
- Trivial Partition: Break the DAG at all multiple fanout points.
  - Guarantees no overlap among trees (no logic replication).
  - Sometimes leads to lots of small trees.

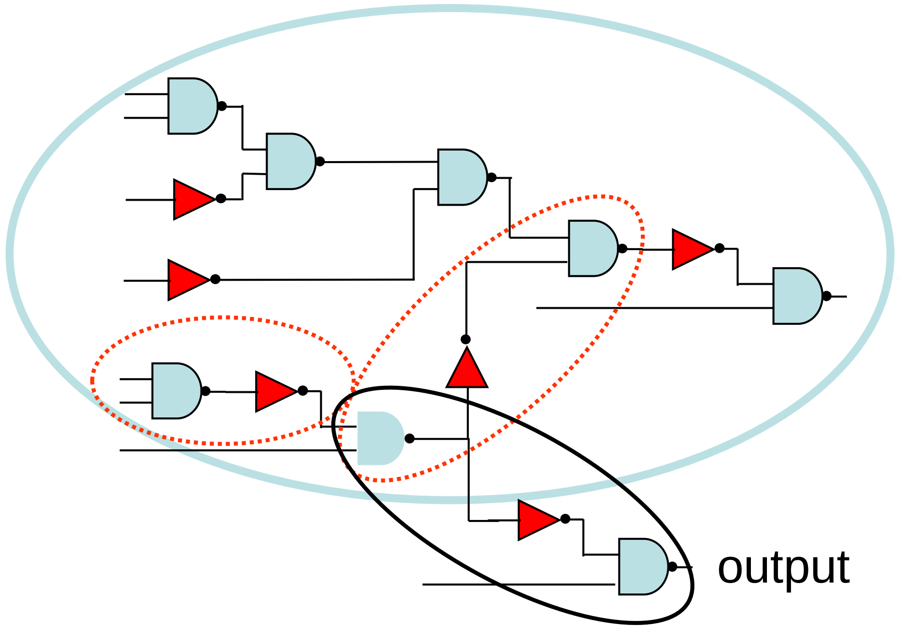




Partitioning a DAG into Trees

- Single-cone partition
  - From a single output, form a large tree back to the primary inputs;
  - Map successive outputs until they hit match output formed from mapping previous primary outputs.
  - Duplicates some logic (where trees overlap)
  - Produces larger trees, better area results in practice





Summary: Technology Mapping

- Three different approaches
  - Rule-based
  - Structural matching
    - Graph covering problem (subject DAG, pattern DAGs)
      - Binate covering formulation
        - » Too slow in practice
      - Tree covering
        - » Dynamic programming, very efficient and scalable
  - Boolean matching

General Principles

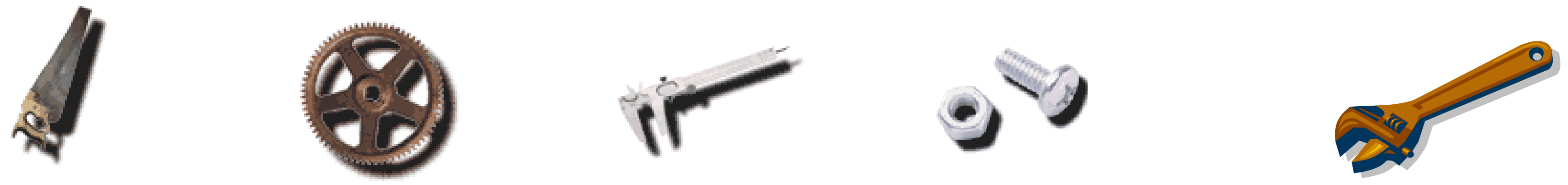
- Look for problems from a different domain that are equivalent to the problem you are trying to solve
  - Technology mapping in logic synthesis ⇔ code generation in compilers
- Combining exact solutions for sub-problems may give an exact (or a good approximate) solution for the complete problem

Suggested Reading

- Hachtel & Somenzi, Chapter 13
- De Micheli, Chapter 10.1,10.2,10.3.1-10.3.3
- Key Papers
  - K. Keutzer, "DAGON: Technology binding and local optimization by DAG matching," in Proc. 24th ACM/IEEE Conf. on Design Automation, pp. 341-347, 1987
  - “MIS: A Multiple-Level Logic Optimization System”, R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, A. R. Wang, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 6, no. 6, Nov. 1987, pp. 1062 - 1081
  - “Logic synthesis for VLSI design”, R. Rudell, Ph.D. thesis, U. C. Berkeley, 1989.
  - Performance-Oriented Technology Mapping, (H.J. Touati, Ch.W. Moon and R. K. Brayton), Proceedings of MIT VLSI Conference, 1990.
  - “Technology mapping for low power”, V. Tiwari, P. Ashar, and S. Malik, Design Automation Conference, pp. 74-79, 1993.

Acknowledgments

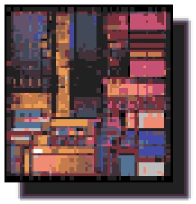
- Prof. Sharad Malik, Princeton
- Prof. Rob Rutenbar, CMU



ECE 595Z
Digital VLSI Design Automation

Module 6 (Lectures 21-24): Timing Analysis and Optimization

Lecture 21



Anand Raghunathan

MSEE 318

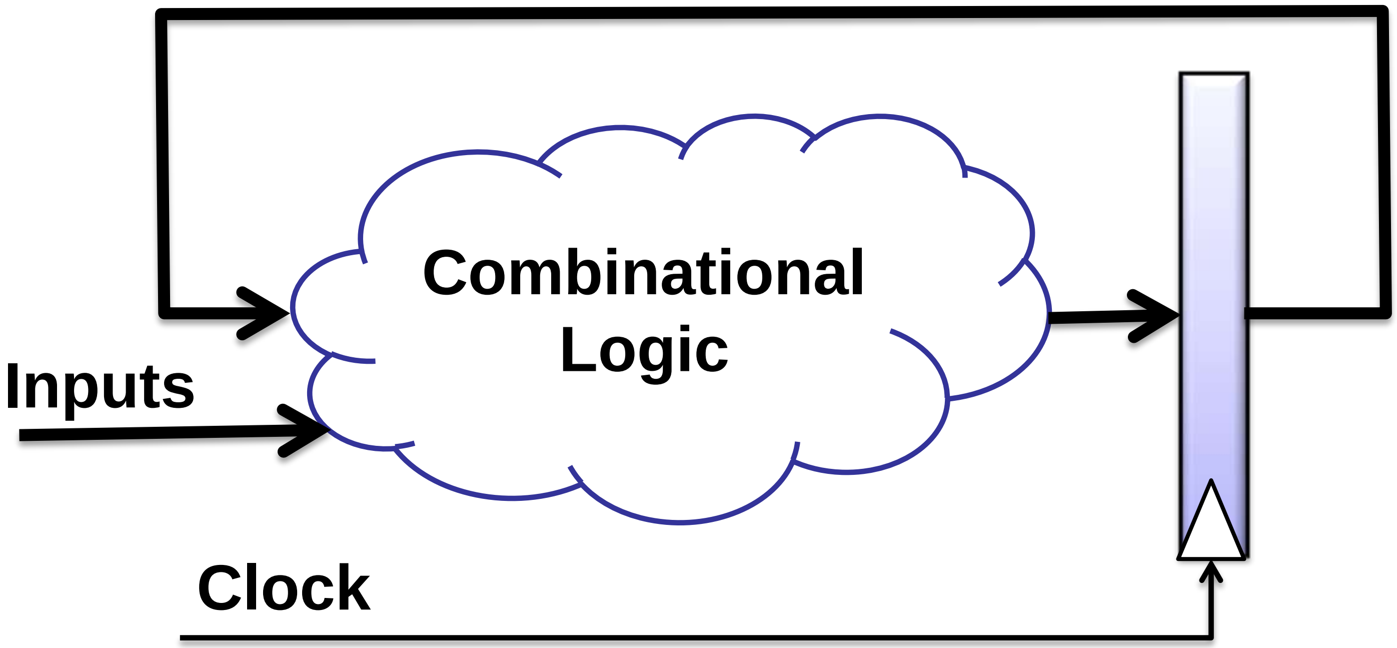
raghunathan@purdue.edu

Timing Optimization

- We have learnt thus far how to synthesize (small) circuits fast
- But, we also want to synthesize FAST circuits!
- Until recently, performance (clock frequency) was undisptued king for most ICs
  - Still is important, except power also matters in most applications and can be the primary metric in some

Timing Analysis

- Given a (sequential) circuit, how fast can I clock it while maintaining correct operation?



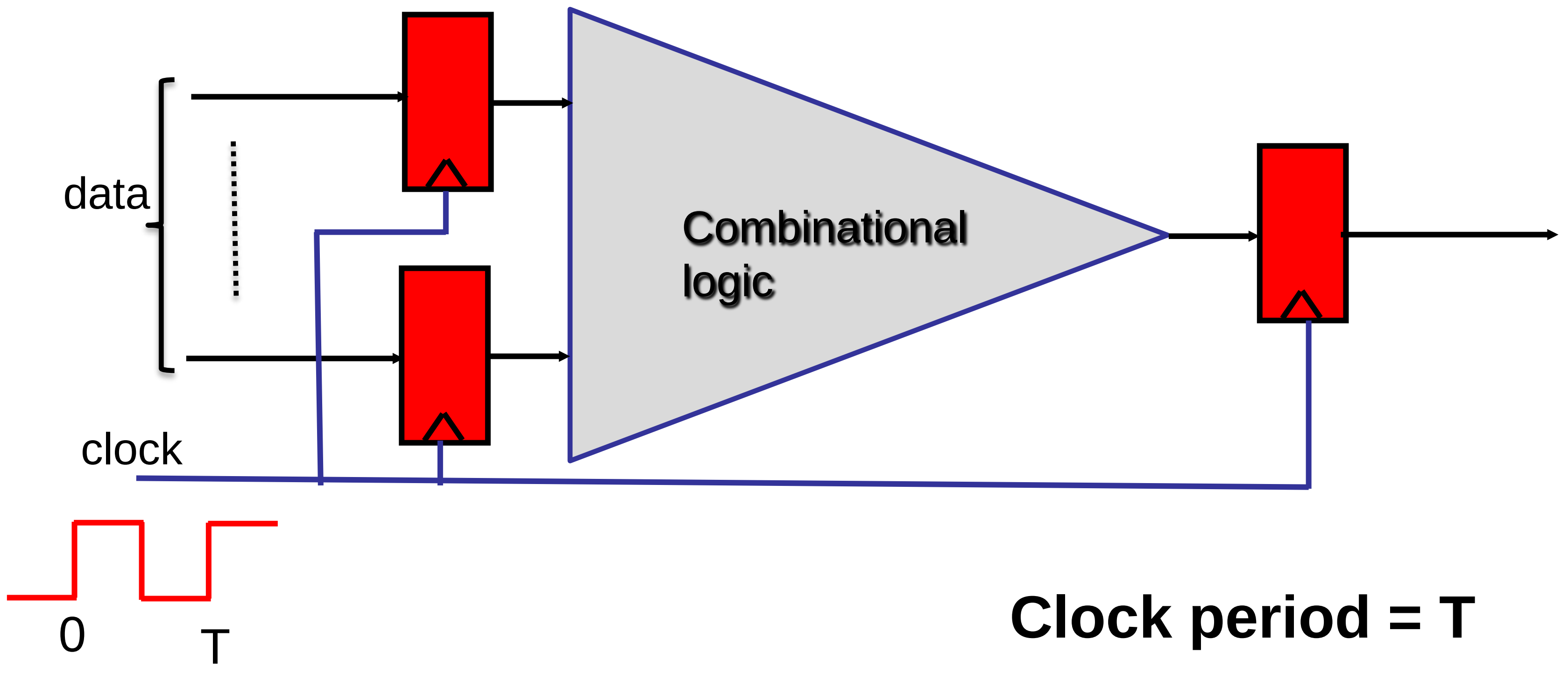
Outline

- Timing Analysis
  - Clocking criteria for sequential circuits
  - Timing graph
  - Delay models for gates
  - Topological timing analysis
  - Functional timing analysis
- Timing Optimization
  - Collapsing and re-structuring
  - Generalized Bypass Transform
  - Generalized Select Transform
  - Eliminating false paths

Timing Analysis

Clocking Criteria for Sequential Circuits

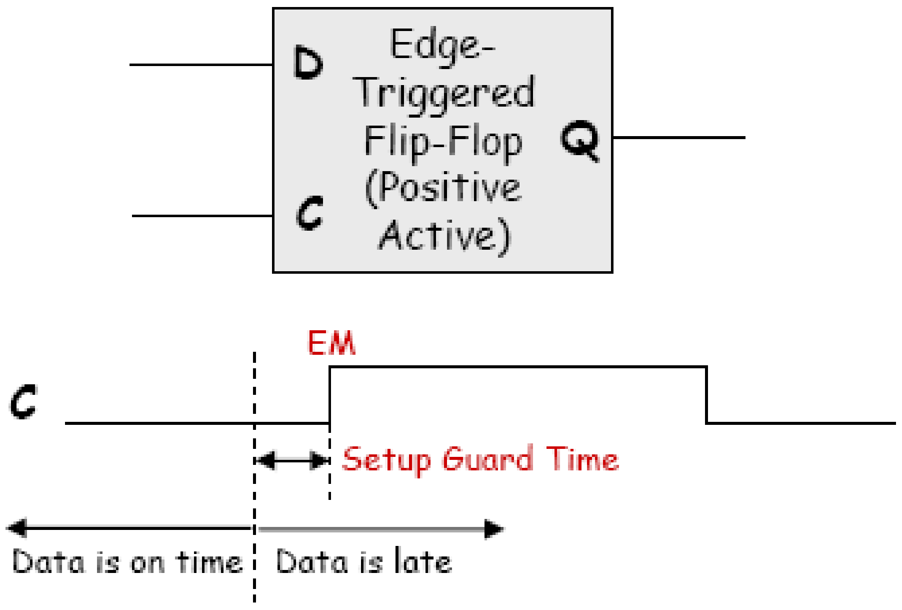
- Consider circuits with edge-triggered storage elements (Flip-Flops)
- What timing properties should be satisfied for correct operation?



Clock period = T

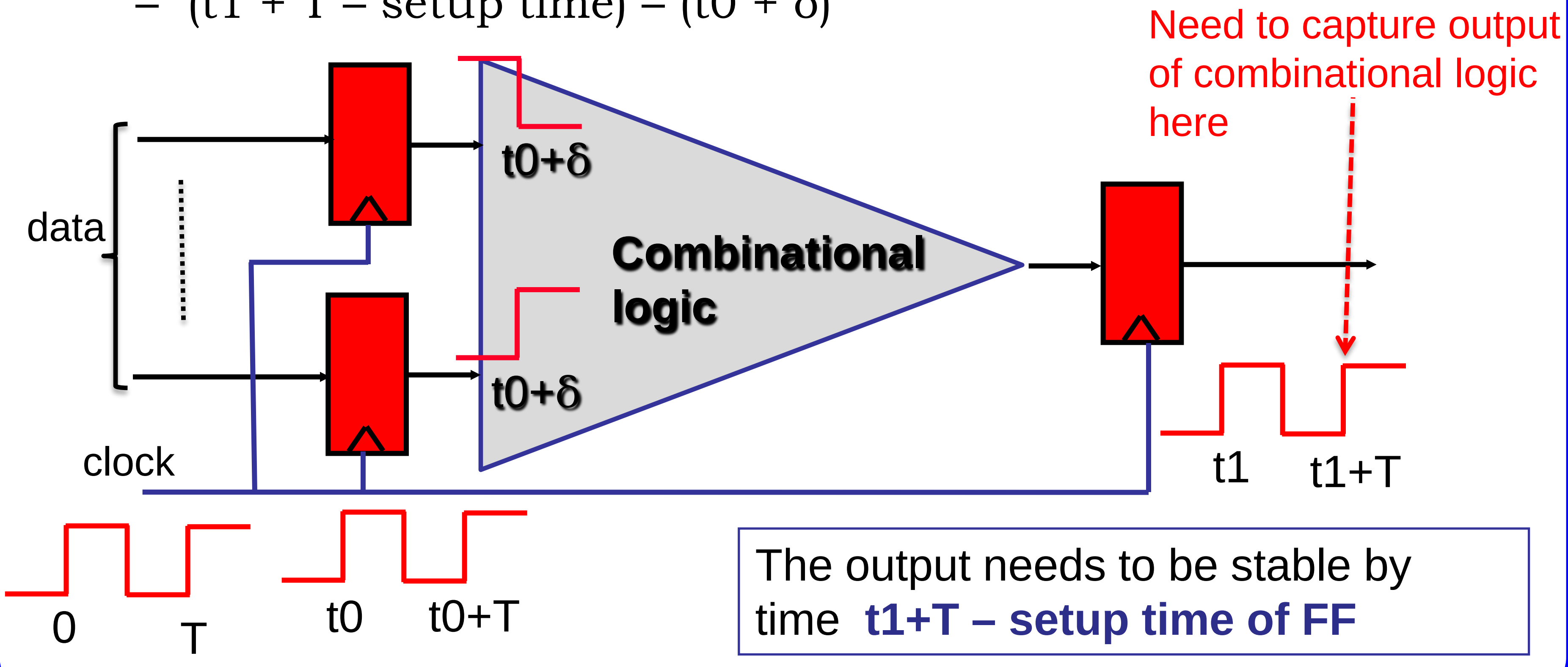
Setup Condition

- Data that we intend to capture at a Flip-Flop must arrive before the clock edge by at least the setup time



Setup Criterion for Sequential Circuits

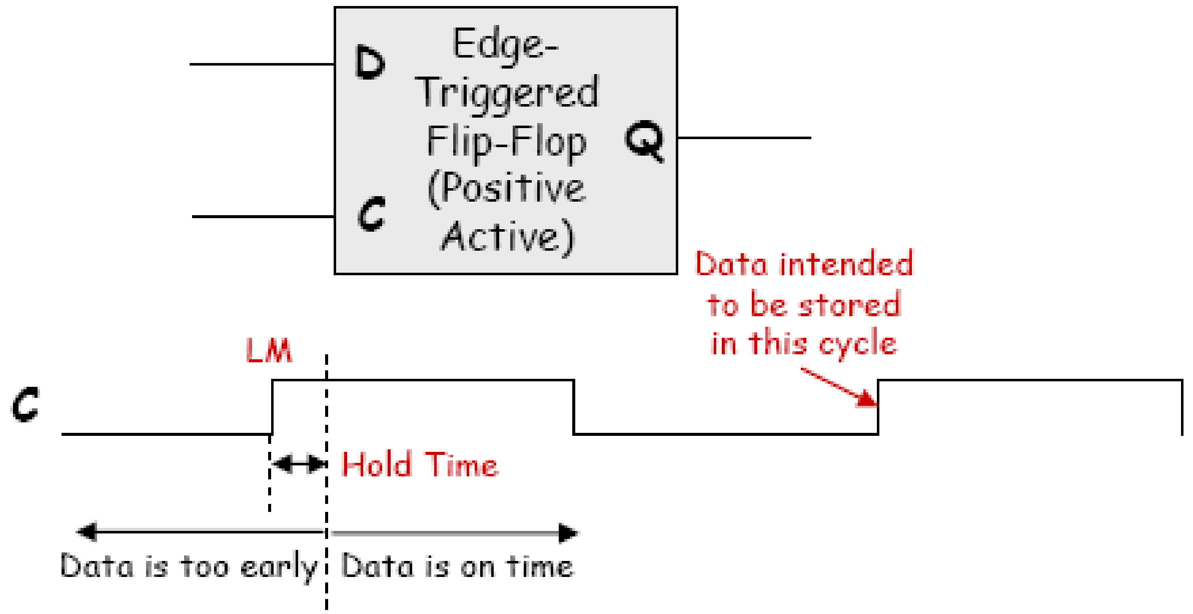
- **Setup condition**: The outputs of the combinational logic should settle in time to be captured
- Translates to an upper bound on the longest delay through the combinational logic
  - (t1 + T – setup time) – (t0 + δ)



The output needs to be stable by time **t1+T – setup time of FF**

Hold Condition

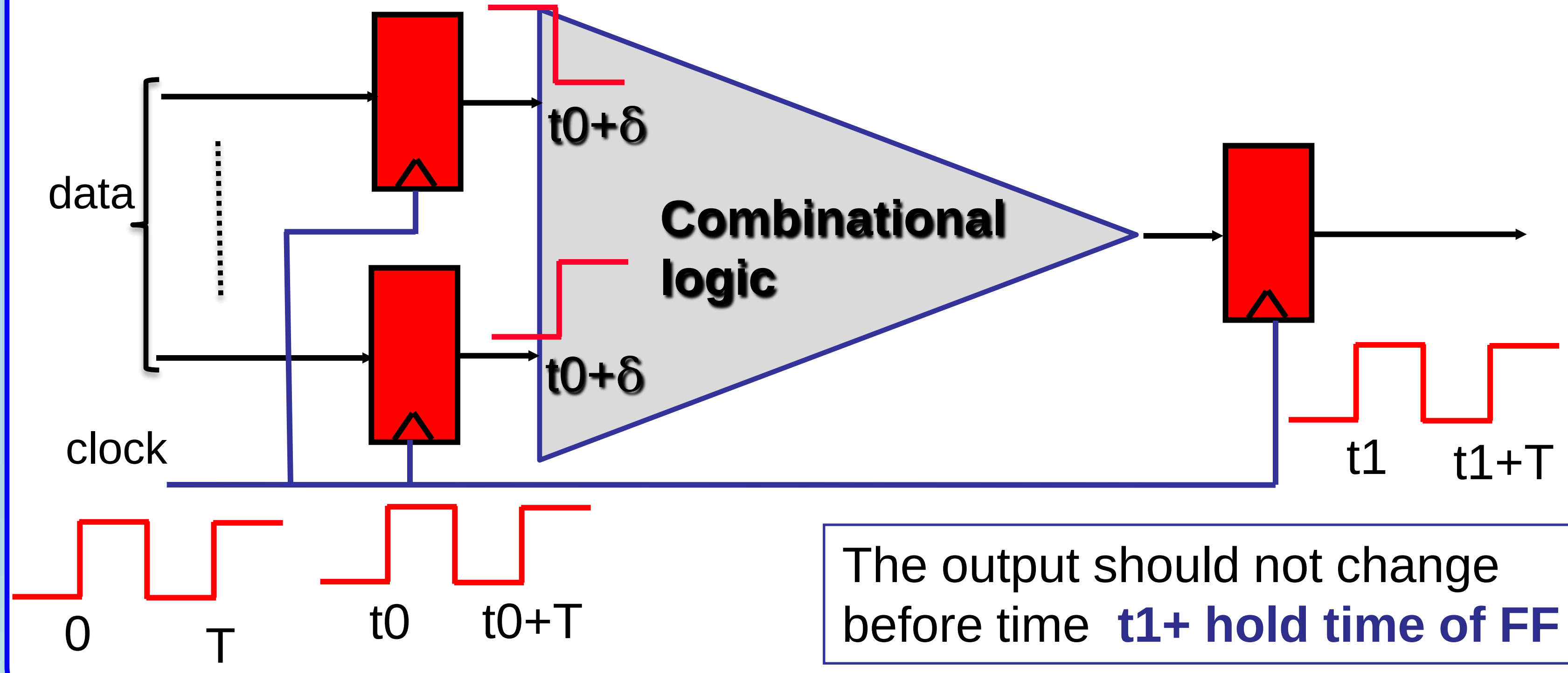
- Data that we intend to capture at a Flip-Flop must not change after the clock edge until at least the hold time



D
Edge-Triggered Flip-Flop (Positive Active)
Q
C
Data intended to be stored in this cycle
LM
C
Hold Time
Data is too early
Data is on time

Hold Criterion for Sequential Circuits

- **Hold condition**: The outputs of the combinational logic should not change too early
- Translates to a lower bound on the shortest delay through the combinational logic
  - (t1 + hold time) – (t0 + δ)



The output should not change before time **t1+ hold time of FF**

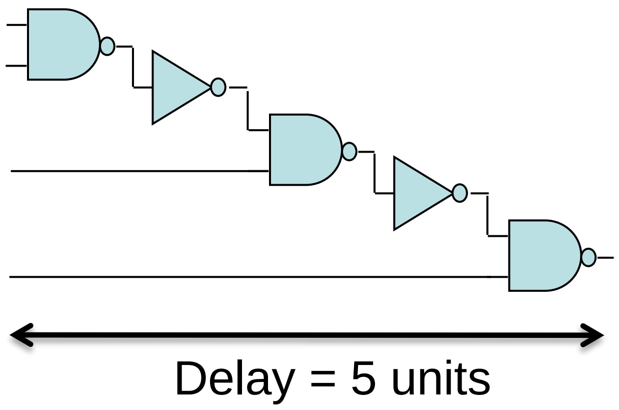
Delay Models for Gates

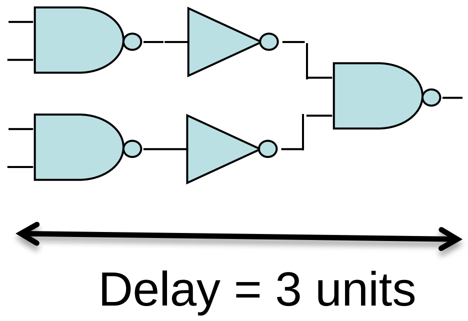
- Unit delay
- Constant delay
- Pin-to-pin delay
- State and Transition dependent delay
- Load and slew rate dependent delay
- PVT corners
- Statistical delay models

Accuracy vs. computation time tradeoff

Unit Delay Model

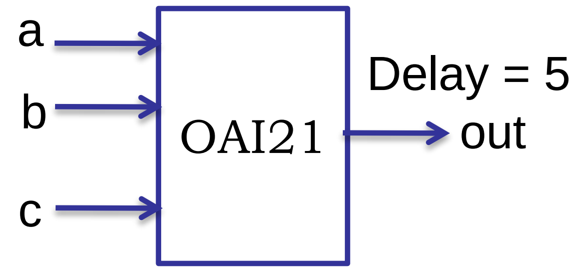
- Simplest model: Assume each gate has a fixed delay of 1 unit
- Usually applied to a network of 2-input gates and inverters
- Typically used in technology-independent optimizations
- Still useful for coarse-grained comparisons between alternative circuit structures

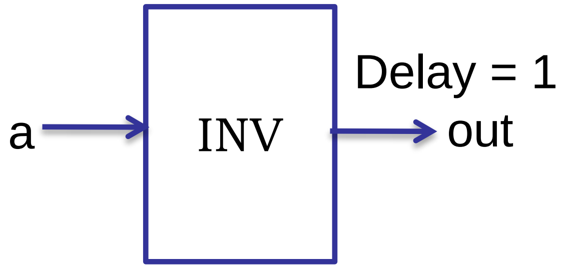




Constant Delay Model

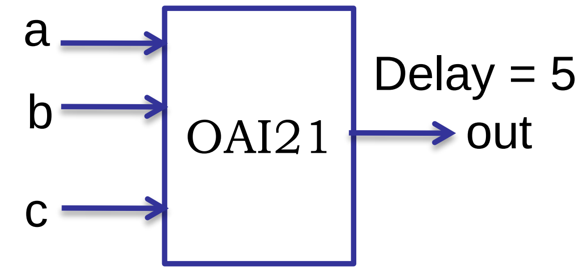
- Different but fixed delay for each gate type
- Simplest technology-dependent delay model

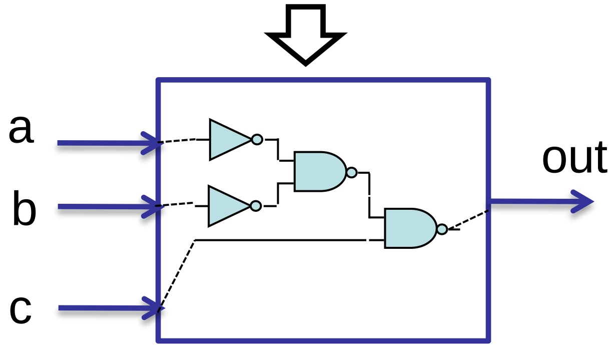


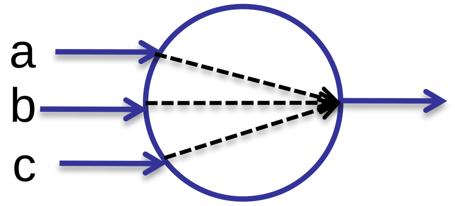


Pin-to-Pin Delay Model

- Not all pins are created equal!
- Accounts for the fact that different paths through a gate can have different delays
- Input to output delay depends on transistor-level implementation of gate/cell



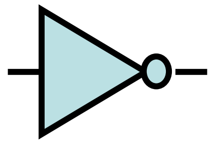




D(a,out) = 5
D(b,out) = 5
D(c,out) = 2

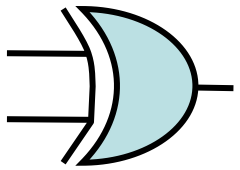
State / Transition Dependent Delay Model

- The values on inputs of the gate actually matter in determining its delay
  - Example: Not all transitions are equal! Rising and falling transition at output have different delays
- Note: Gate delay can be dependent on history



Delay(o/p rising) = 1.2
Delay(o/p falling) = 1

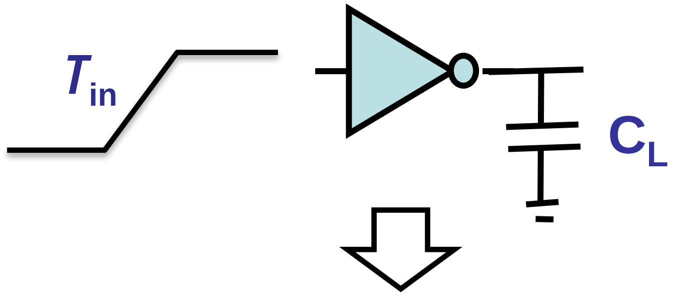
XOR2



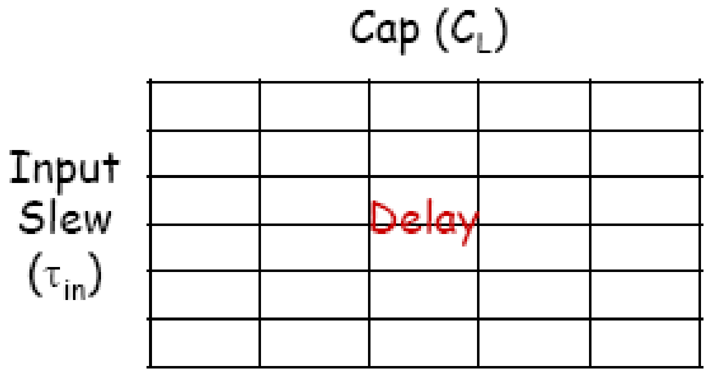
Delay(i/p falling, o/p falling) = 3.1
Delay(i/p falling, o/p rising) = 3.4
Delay(i/p rising, o/p falling) = 3.6
Delay(i/p rising, o/p rising) = 3.8

Load and Slew Rate Dependent Model

- Considers dependency of delay on output load and input slew rate
  - Lookup table with inter-polation
    - Discretize useful range of input slew rates and output loads
  - Equation
    - Fit simulated / measured data
- Need similar model for computing slew rate at output



Delay table (o/p falling)

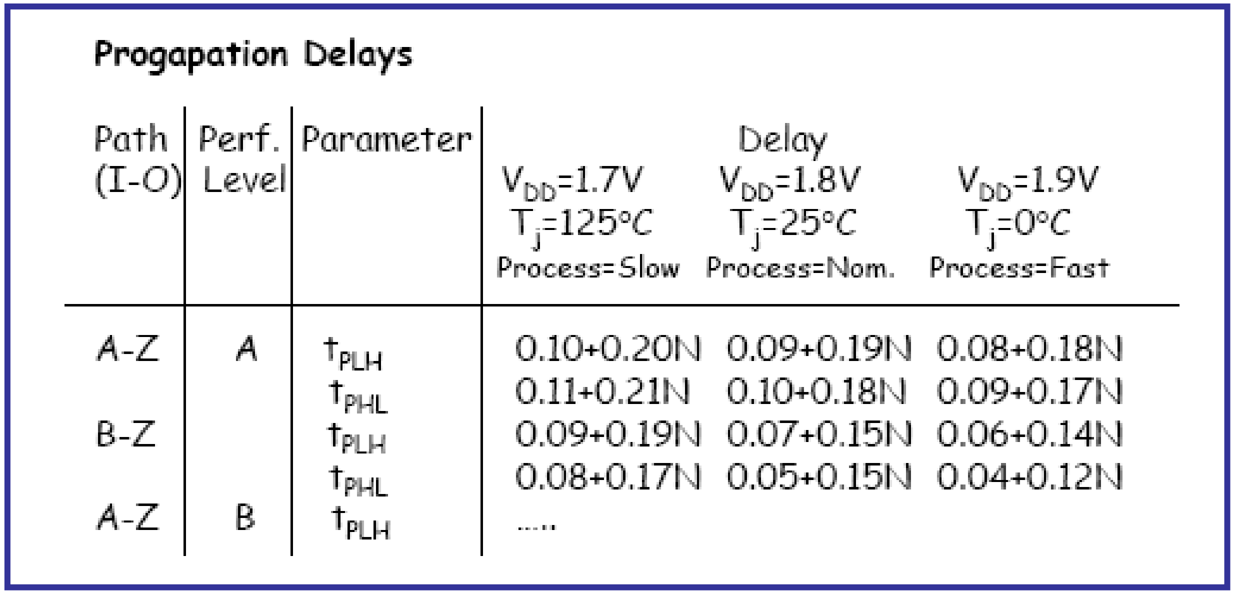


Delay equation (o/p falling)

$$\text{Delay} = \alpha \cdot \tau_{in} + \beta \cdot C_L + \gamma \cdot \tau_{in} \cdot C_L + \delta$$

PVT Corners

- Delay is impacted by Process, Voltage, and Temperature variations
- Conventional approach: Consider "corners"
  - Slow, Typical (or Nominal), Fast
- Problem: Increasing spread leads to very conservative estimates
  - Possible solution: Statistical models (active area of research)



Progapation Delays

Path (I-O)	Perf. Level	Parameter	Delay V_{DD}=1.7V T_j=125°C Process=Slow	V_{DD}=1.8V T_j=25°C Process=Nom.	V_{DD}=1.9V T_j=0°C Process=Fast
A-Z	A	t_{PLH}	0.10+0.20N	0.09+0.19N	0.08+0.18N
		t_{PHL}	0.11+0.21N	0.10+0.18N	0.09+0.17N
B-Z		t_{PLH}	0.09+0.19N	0.07+0.15N	0.06+0.14N
		t_{PHL}	0.08+0.17N	0.05+0.15N	0.04+0.12N
A-Z	B	t_{PLH}	.....		

Summary

- Basic questions in timing analysis
  - Do the outputs of the combinational logic always reach the (final) stable value in time to be correctly captured?
    - Setup condition
  - Do the outputs of the combinational logic stay stable long enough to be correctly captured?
    - Hold condition
- Various delay models possible for gates
- Need delay models for wires too, but we will not talk about them in this class