ECE260B – CSE241A
Winter 2004

Clocking

Website:  http://vlsicad.ucsd.edu/courses/ece260b-w04
State is stored in registers (flip-flops or latches)
Combinational logic computes next-state, outputs from present-state, inputs
Outline

- Why Clocking
- Clock Distribution
- The “Zero-Skew Tree” Problem
- The “Useful” Skew Problem
- The Timing Analysis Problem
Why Clocks?

- Clocks provide the means to synchronize
  - By allowing events to happen at known timing boundaries, we can sequence these events
- Greatly simplifies building of state machines
- No need to worry about variable delay through combinational logic (CL)
  - All signals delayed until clock edge (clock imposes the worst case delay)

![Diagram](image-url)
Clock Cycle Time

- Cycle time is determined by the delay through the CL
  - Signal must arrive before the latching edge
  - If too late, it waits until the next cycle
    - Synchronization and sequential order becomes incorrect

- Constraint: $t_{cycle} > t_{prop\_delay\_through\_CL} + t_{overhead}$
  - Example: 3.0 GHz Pentium-4 $\rightarrow$ $t_{cycle} = 333$ps

- Can change circuit architecture to obtain smaller $T_{cycle}$
**Pipelining**

- **For dataflow:**
  - Instead of a long critical path, split the critical path into chunks
  - Insert registers to store intermediate results
  - This allows 2 waves of data to coexist within the CL

- **Can we extend this ad infinitum?**
  - Overhead eventually limits the pipelining
    - E.g., 1.5 to 2 gate delays for latch or FF
  - Granularity limits as well
    - Minimum time quantum: delay of a gate

\[
t_{\text{cycle}} > t_{\text{pd}} + t_{\text{overhead}}
\]

\[
t_{\text{cycle}} > \max(t_{\text{pd1}}, t_{\text{pd2}}) + t_{\text{overhead}}
\]
- FO4 INV = inverter driving 4 identical inverters (no interconnect)
- Half of frequency improvement has been from reduced logic stages, i.e., pipelining
Let’s Revisit Cycle Time and Path Delay

- Cycle time ($T$) cannot be smaller than longest path delay ($T_{\text{max}}$)
- Longest (critical) path delay is a function of:
  - Total gate, wire delays
    - logic levels

![Diagram of a circuit with gates and wires, showing the critical path and cycle time]

$T_{\text{max}} \leq T$

Courtesy K. Keutzer et al. UCB
**Cycle Time - Setup Time**

- For FFs to correctly capture data, must be stable for:
  - Setup time \( (T_{\text{setup}}) \) *before* clock arrives

\[
T_{\text{max}} + T_{\text{setup}} \leq T
\]

![Diagram of FFs and clock signal](image)

"Critical path, ~5 logic levels"
Cycle Time – Clock Skew

- If clock network has unbalanced delay – clock skew

- Cycle time is also a function of clock skew \( T_{\text{skew}} \)

\[
T_{\text{max}} + T_{\text{setup}} + T_{\text{skew}} \leq T
\]

Courtesy K. Keutzer et al. UCB
Cycle Time – Flip-Flop Delay (Clock to Q)

- Cycle time is also a function of propagation delay of FF ($T_{\text{clk-to-Q}}$ or $T_{c2q}$)

- $T_{c2q}$: time from arrival of clock signal till change at FF output

$$T_{\text{max}} + T_{\text{setup}} + T_{\text{skew}} + T_{\text{clk-to-Q}} \leq T$$
**Min Path Delay - Hold Time**

- For FFs to correctly latch data, data must be stable during:
  - Hold time ($T_{\text{hold}}$) *after* clock arrives
  - Determined by delay of shortest path in circuit ($T_{\text{min}}$) and clock skew ($T_{\text{skew}}$)

\[
T_{\text{min}} \geq T_{\text{hold}} + T_{\text{skew}}
\]

- Short path, $\sim 3$ logic levels

---

ECE 260B – CSE 241A Clocking 12

Courtesy K. Keutzer et al. UCB

Andrew B. Kahng, UCSD
Setup, Hold, Cycle Times

- **cycle time**

- **hold time** – D stable after clock

- **set-up time** – D stable before clock

Example of a single phase clock

When signal may change

Courtesy K. Keutzer et al. UCB
Summary of Constraints (Edge-Triggered FFs)

- $\text{Max}(t_{pd}) < t_{per} - t_{su} - t_{c2q} - t_{skew}$
  - Delay is too long for data to be captured

- $\text{Min}(t_{pd}) > t_h - t_{c2q} + t_{skew}$
  - Delay is too short and data can race through, skipping a state

Courtesy K. Yang, UCLA
Example of $t_{pd\max}$ Violation

- Suppose there is skew between the registers in a dataflow (regA after regB)
- “i” gets its input values from regA at transition in $Ck'$
- CL output “o” arrives after $Ck$ transition due to skew
- To correct this problem, can *increase* cycle time
Example of $t_{\text{pdmin}}$ Violation: Race Through

- Suppose clock skew causes regA to be clocked before regB
- “i” passes through the CL with little delay ($t_{\text{pdmin}}$)
- “o” arrives before the rising $Ck'$ causes the data to be latched
- **Cannot be fixed by changing frequency** → have rock instead of chip

![Diagram showing timing and skew](https://via.placeholder.com/150)
Time Borrowing (Cycle Stealing, Useful Skew)

- Cycle steal with flip-flops using delayed clocks
Outline

- Why Clocking
- Clock Distribution
- The “Zero-Skew Tree” Problem
- The “Useful” Skew Problem
- The Timing Analysis Problem
Clock Distribution

- General goal of clock distribution
  - Deliver clock to all memory elements with acceptable skew
  - Deliver clock edges with acceptable sharpness

- Clocking network design is one of the greatest challenges in the design of a large chip

- Clocks generally distributed via wiring trees (and meshes)

- Low-resistance interconnect to minimize delay

- Multiple drivers to distribute driver requirements
  - Use optimal sizing principles to design buffers
  - Clock lines can create significant crosstalk
Clock Distribution Problem Statement

- **Objective**
  - Minimum skew (performance and hold time issues)
  - Minimum cell area and metal use
  - (sometimes) minimal latency
  - (sometimes) particular latency
  - (sometimes) intermixed gating for power reduction
  - (sometimes) hold to particular duty cycle: e.g. 50:50 +- 1 percent

- **Subject to:**
  - Process variation from lot-to-lot
  - Process variation across the die
  - Radically different loading (ff density) around the die
  - Metal variation across the die
  - Power variation across the die (both static IR and dynamic)
  - Coupling (same and other layers)
Issues in Clock Distribution Network Design

- Skew
  - Process, voltage, and temperature
  - Data dependence
  - Noise coupling
  - Load balancing

- Power, $CV^2f$
  - Clock gating

- Flexibility/Tunability
  - Compactness – fit into existing layout/design

- Reliability
  - Electromigration
Skew: Clock Delay Varies With Position
Clock Skew Causes

- Designed (unavoidable) variations – mismatch in buffer load sizes, interconnect lengths
- Process variation – process spread across die yielding different \( L_{\text{eff}}, T_{\text{ox}}, \) etc. values
- Temperature gradients – changes MOSFET performance across die
- IR voltage drop in power supply – changes MOSFET performance across die
- Note: Delay from clock generator to fan-out points (clock latency) is not important by itself
  - BUT: increased latency leads to larger skew for same amount of relative variation

Sylvester / Shepard, 2001
Clock Distribution Methods

- **RC-Tree**
  - Less capacitance
  - More accuracy
  - Flexible wiring

- **Grids**
  - Reliable
  - Less data dependency
  - Tunable (late in design)

Shown here for final stage drivers driving F/F loads
Grids

- Gridded clock distribution common on earlier DEC Alpha microprocessors

**Advantages:**
- Skew determined by grid density, not too sensitive to load position
- Clock signals available everywhere
- Tolerant to process variations
- Usually yields extremely low skew values

**Disadvantages:**
- Huge amount of wiring and power
- To minimize such penalties, need to make grid pitch coarser $\rightarrow$ lose the grid advantage

Sylvester / Shepard, 2001
H-Tree

- H-tree (Bakoglu)
  - One large central driver, recursive structure to match wirelengths
  - Halve wire width at branching points to reduce reflections

- Disadvantages
  - Slew degradation along long RC paths
  - Unrealistically large central driver
    - Clock drivers can create large temperature gradients (ex. Alpha 21064 ~30\(^\circ\) C)
  - Non-uniform load distribution
  - Inherently non-scalable (wire R growth)
  - Partial solution: intermediate buffers at branching points

courtesy of P. Zarkesh-Ha
Buffered H-tree

- Advantages
  - Ideally zero-skew
  - Can be low power (depending on skew requirements)
  - Low area (silicon and wiring)
  - CAD tool friendly (regular)

- Disadvantages
  - Sensitive to process variations
    - Devices → Want same size buffers at each level of tree
    - Wires → Want similar segment lengths on each layer in each source-sink path !!!
  - Local clocking loads inherently non-uniform
Tree Balancing

Some techniques:

a) Introduce dummy loads

b) Snaking of wirelength to match delays

Con: Routing area often more valuable than Silicon
Examples From Processor Chips

H-Tree, Asymmetric
RC-Tree (IBM)

Grids
DEC [Alphas]

Serpentines
Intel x86
[Young ISSCC97]
Example Skews From Processor Chips

DEC-Alpha 21064 clock spines

DEC-Alpha 21064 RC delays

DEC-Alpha 21164 RC delays for Global Distribution (Spine + Grid)

DEC-Alpha 21164 RC local delays
ReShape Clocks Example (High-End ASIC)

- Balanced, shielded H-tree for pre-clock distribution
- Mesh for block level distribution

- All routes 5-6u M6/5, shielded with 1u grounds
- ~10 buffers per node
  - E.g., ganged BUFx20’s
- Output mesh must hit every sub-block
Block Level Mesh (.18u)

- Clumps of 1-6 clock buffers, surrounded by capacitor pads
- Shielded input and output m6 shorting straps
- Pre-clock connects to input shorting straps
- 1u m5 ribs every 20 - 30 u (4 to 6 rows)
- Max 600u stride
Problems with Meshes

- Burn more power at low frequencies
- Blocks more routing resources (solution: integrated power distribution with ribs can provide shielding for ‘free’)
- Difficult for ‘spare’ clock domains that will not tolerate regioning
- Post placement (and routing) tuning required
- No ‘beneficial skew’ possible
- Clock gating only easy at root

Fighting tools to do analysis:
- Clumped buffers a problem in Static Timing Analysis tools
- Large shorted meshes a problem for STA tools
- What does Elmore delay calculation look like for a non-tree?
- Need full extraction and SPICE-like simulation to determine skew
Benefits of Meshes

- Deterministic since shielded all the way down to rib distribution
- No ECO placement required: all buffers preplaced before block placement
- Low latency since uses shorted (= ganged, parallel) drivers, therefore lower skew
- ECO placements of FFs later do not require rebalancing of tree
- “Idealized” clocking environment for “concurrent dance” of RTL design and timing convergence
Outline

- Why Clocking
- Clock Distribution
- The “Zero-Skew Tree” Problem
- The “Useful” Skew Problem
- The Timing Analysis Problem
Zero-Skew Tree (ZST) Problem

- Zero Skew Clock Routing Problem \((S,G)\): Given a set \(S\) of sink locations and a connection topology \(G\), construct a ZST \(T(S)\) with topology \(G\) and having minimum cost.

- Skew = maximum value of \(|t_d(s_0,s_i) - t_d(s_0,s_j)|\) over all sink pairs \(s_i, s_j\) in \(S\).

- \(T_d = \) signal delay (from source \(s_0\))

- Connection topology \(G\) = rooted binary tree with nodes of \(S\) as leaves
  - Edge \(e_a\) in \(G\) is the edge from \(a\) to its parent
  - \(|e_a|\) is the (assigned) length of edge \(e_a\)

- Cost = total edge length
Zero-Skew Example (555 sinks, 40 obstacles)
A Zero-Skew Routing Algorithm

- Finds a ZST under linear delay model with minimum cost over all ZSTs with topology G and sink set S

- Terms
  - Manhattan Arc: line segment with slope +1 or -1
  - Tilted Rectangular Region (TRR): collection of points within a fixed distance of a Manhattan arc
    - Core = Manhattan arc
    - Radius = distance
  - Merging segment = locus of feasible locations for a node v in the topology, consistent with minimum wirelength
    - If v is a sink, then ms(v) = {v}
    - If v is an internal node, then ms(v) is the set of all points within distance |e_a| of ms(a), and within distance |e_b| of ms(b)
Phase 1: Tree of Merging Segments

- Goal: Construct a tree of merging segments corresponding to topology G
  - Merging segment of a node depends on merging segment of its children → **bottom-up construction**
  - Let a, b be children of v. We want placements of v that allow TS_a and TS_b to be merged with minimum added wire while preserving zero skew
    - Merging cost = |e_a| + |e_b|

- Fact: The intersection of two TRRs is also a TRR and can be found in constant time

- Constant time per each new merging segment → linear time (in size of S) to construct entire tree
Phase 2: Find Node Placements

- Goal: Find exact locations ("embeddings") \( \text{pl}(v) \) of internal nodes \( v \) in the ZST topology

- If \( v \) is the root node, then any point on \( \text{ms}(v) \) can be chosen as \( \text{pl}(v) \)

- If \( v \) is an internal node other than the root, and \( p \) is the parent of \( v \), then \( v \) can be embedded at any point in \( \text{ms}(v) \) that is at distance \( |e_v| \) or less from \( \text{pl}(p) \)
  - Detail: create square TRR \( \text{trr}_p \) with radius \( e_v \) and core equal to \( \text{pl}(p) \); placement of \( v \) can be any point in \( \text{ms}(v) \cap \text{trr}_p \)

- Each instruction executed at most once for each node in \( G \), and TRR intersection is \( O(1) \) time \( \rightarrow \) Find_Exact_Placements is \( O(n) \) \( \rightarrow \) DME is \( O(n) \)
Non-Zero Skew Bounds

- Given a skew bound, where can internal nodes of the given topology (e.g., a, b, v) be placed?
BST-DME Bottom-Up Phase

Bottom-Up: build tree of merging regions corresponding to given topology

$B = 4$

$mr(a)$

$mr(v)$

$mr(b)$
BST-DME Top-Down Phase

Topology

B = 4
Outline

- Why Clocking
- Clock Distribution
- The “Zero-Skew Tree” Problem
- The “Useful” Skew Problem
- The Timing Analysis Problem
Skew = Local Constraint

- Timing is correct as long as the clock signals of sequentially adjacent FFs arrive within a permissible skew range.

\[-d + t_{\text{hold}} < \text{Skew} < T_{\text{period}} - D - t_{\text{setup}}\]

race condition \hspace{1cm} cycle time violation

permissible range
“Useful Skew” → Design Robustness

- Design will be more robust if clock signal arrival time is in the middle of permissible skew range, rather than on edge.
Constraints on Skews

- $\text{FF}_i$ receives clock signal delayed by $x_i \geq \text{MIN}_\text{DEL}$
  - $0 < \alpha \leq 1 \leq \beta$: if nominal clock delay is $x_i$, then actual clock delay must fall within interval $\alpha x_i \leq x \leq \beta x_i$
  - For FF to operate correctly when clock edge arrives at time $x$, the correct input data must be present and stable during the time interval $(x - \text{SETUP}, x + \text{HOLD})$
  - For $1 \leq i,j \leq L$ (#FFs), we compute lower and upper bounds $\text{MIN}(i,j)$ and $\text{MAX}(i,j)$ for the time that is required for a signal edge to propagate from $\text{FF}_i$ to $\text{FF}_j$

- Avoid double-clocking (race condition)
  - $\alpha x_i + \text{MIN}(i,j) \geq \beta x_j + \text{HOLD}$

- Avoid zero-clocking
  - $\beta x_j + \text{SETUP} + \text{MAX}(i,j) \leq \alpha x_j + P$; $P = \text{clock period}$
Optimal Useful Skews by Linear Programming

- **LP_SPEED (clock period reduction):**
  
  minimize $P$ s.t.
  
  $\alpha x_j - \beta x_j \geq HOLD - MIN(i,j)$
  
  $\alpha x_i - \beta x_j + P \geq SETUP + MAX(i,j)$
  
  $x_i \geq MIN_{DEL}$

- **LP_SAFETY (robustness):**
  
  Maximize $M$ s.t.
  
  $\alpha x_j - \beta x_j - M \geq HOLD - MIN(i,j)$
  
  $\alpha x_i - \beta x_j - M \geq SETUP + MAX(i,j) - P$
  
  $x_i \geq MIN_{DEL}$

- **Notes**
  
  - Useful Skew optimization is similar to Retiming optimization
  - Peak current reductions are a side benefit
Outline

- Why Clocking
- Clock Distribution
- The “Zero-Skew Tree” Problem
- The “Useful” Skew Problem
- The Timing Analysis Problem
Is Circuit Timing Correct?

original circuit

extracted block
Key = Delays Through Combinational Block

- Arrival time in green
- Interconnect delay in red
- Gate delay in blue

Question: What is the right mathematical object to use to represent this physical object?

Courtesy K. Keutzer et al. UCB
**Actual Arrival Time**

- Actual arrival time $A(v)$ for a node $v$ is **latest** time that signal can arrive at node $v$

\[
A(v) = \max_{u \in Fl(v)} (A(u) + d_{u \rightarrow v})
\]

where $d_{v \rightarrow u}$ is delay from $v$ to $u$, $Fl(u) = \{X, Y\}$, and $v = \{Z\}$.
Problem: Longest Path in Directed Graph

- Use a labeled directed graph
- \( G = <V, E> \)
- Vertices represent gates, primary inputs and primary outputs
- Edges represent wires
- Labels represent delays