Lecture 8:

Skew Tolerant Design (including Dynamic Circuit Issues)

Computer Systems Laboratory
Stanford University
horowitz@stanford.edu

Introduction

- Reading (for Monday’s lecture on Transistors)
  - – Chen Predicting CMOS Speed
  - – Pelgrom Transistor Matching
  - – Lovett Transistor Matching

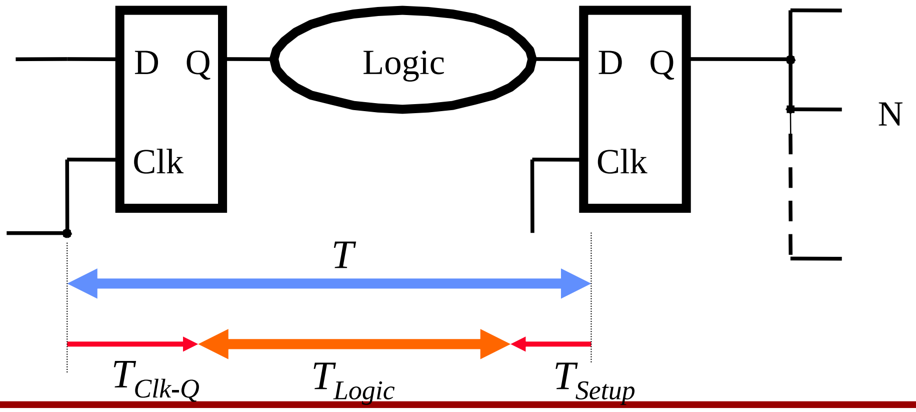
- Overview
  - – The previous lectures talked about clocked storage elements (flops and latches). This lecture will look at the implications of these issues, and circuit approaches for minimizing clocking overheads. We will also look at how clocking issues affect domino circuits.

Just Remember

- Design of clocking system is critical for all modern circuits
  - There are many ways of messing up the design
    - If you do, nothing works, so the chip is dead
    - New silicon costs the company 3 months for fabrication, and probably $1M dollars
  - As a designer you generally don’t set the clocking method
    - Follow the scheme set up by senior designs
    - Rules are generally pretty rigid
- Why so critical?
  - Latches or Flops used everywhere in the design
  - Min delay failures mean the chip does not work
  - Most designs think about max path issues.

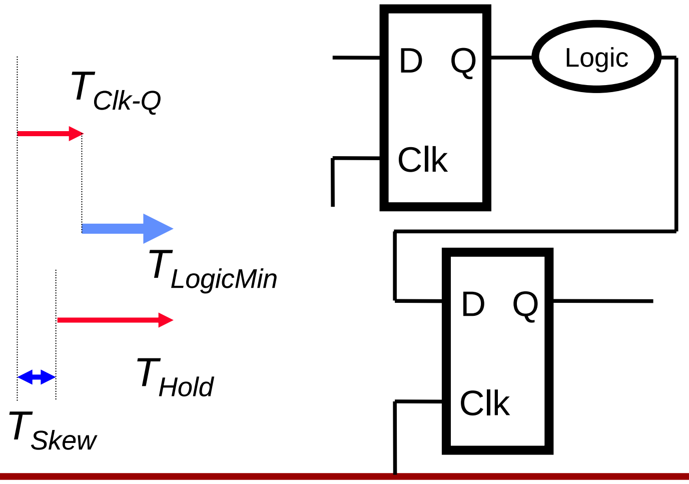
Max Path Constraint

- What most people worry about, since it affects performance
  - $T_{cyc} > T_{Clk\text{-}Q} + T_{Logic} + T_{Setup} + T_{Skew} + T_{jitter}$
- The smallest T you will see between the flops is
  - $T_{cyc} - (T_{Skew} + T_{jitter})$



Min Path Constraint

- Need to ensure that the new data does not arrive to soon
  - $T_{LogicMin} > T_{Skew} + T_{Hold} - T_{Clk\text{-}Q}$



Max Path vs. Min Path Constraints

- Max path is not the most important problem
  - It does set performance
    - But can make the design work by making Tcyc longer
  - Design margins for skew and jitter are for expected values

- Min path is
  - If you have it wrong, your chip does not work
    - Changing the freq will not help
  - Design margins for skew for min path must be worst-case
    - These are MUCH worse than expected values

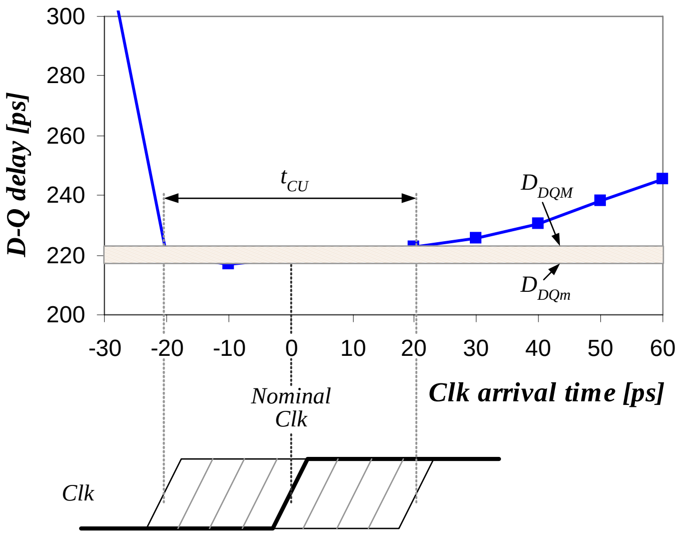
Skew Tolerant Design

- Performance and function of design not sensitive to skew
  - Need to remove skew from max path equation
  - At the same time we don’t create min path problems
  - This combination is hard!

- The basic problem:
  - For easiest min path you want long $T_{Clk\text{-}Q}$
    - And T_{Hold} to be as small as possible
      - Min T_{Hold} must be larger than $(-T_{Setup})$
      - So want T_{Setup} to be positive
  - For max performance you want short $T_{Clk\text{-}Q}$ and negative T_{Setup}
    - Which of course is the worse situation for min path issues

Transparency and Setup Hold Windows

- The transparency window is good for avoiding skew effects

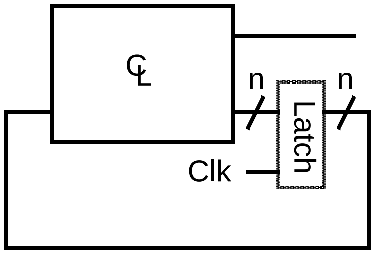


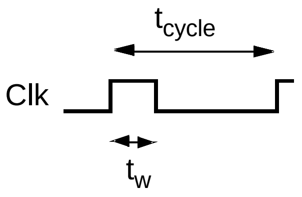
Skew of +/- 20ps will not affect the output timing of this flop (D-Q delay is unchanged)

Pulse Model Latch/Flop Design

- Timing rules are the same as before
  - $t_{dmax} < t_{cycle} - t_{setup} - t_{clk\text{-}q} – t_{skew} - t_{jitter}$
  - $t_{dmin} > t_{skew} + t_{hold} - t_{clk\text{-}q}$
- But the flop parameters are different
  - $t_{clk\text{-}q}$ is smaller since it is a single latch
  - t_{setup} can be negative, by roughly the pulse width
- The hold time is now a large positive number!

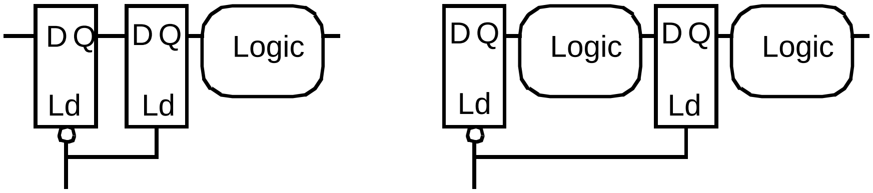
You need to be careful when the setup time is negative, since you can get a situation where the max delay can be longer than a cycle. This is in fact the case, and is called timing borrowing, since you are borrowing from the previous cycle, but you need to make sure all data loops (when you get back to your starting gate take one cycle. We will talk about this a little later





Latch Based Design

- Can you get soft boundaries without making hold time terrible?
  - Yes, kind of
- Why did MS Flops have two latches?
  - Want to make a very short acquisition pulse
  - So make it the series of two pulses (one for each latch)
  - If the pulses don’t overlap, effective pulse width is negative
- Break flop into its two latches, use clock as the “pulse”
  - Place logic between latches



Latch Based Clocking

- Every cycle is broken by two latches
  - That means that each signal must go through two latches
  - So if you set clocks up correctly, hold time should not be bad
    - Problem is each latch has a different clock, so skew can cause hold time issues
- But in this system there are no hard edges
  - Transparency window of each latch is large, ½ a clock cycle

- The large transparency window means
  - Can borrow time naturally
    - Can have up to 1.5 cycle (if there was no skew) in some cycle, if the adjacent cycle only needs .5 w/o skewing clocks
  - Is insensitive to clock skew; for critical paths, data sets timing

Thinking About Timing

- When you can borrow time, thinking about timing becomes confusing
  - You don't really know when the output transitions; it depends on input
  - And, of course, the input timing depends on the output
- At these times, remember that what really matters are logic cycles
  - Data has to arrive back when you assumed it would arrive.

- Image your arranging your netlist on a sheet; place all the flops at the top
  - The gates distance from the top indicates the settling time of its output
  - Gates at the end of long paths would be at the bottom of the sheet
  - Some of the outputs are the inputs to the flops, so we roll the sheet
  - Forms a cylinder, where the circumference is equal to the cycle time

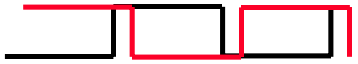
- With flops the problem is simple, since the timing of the outputs is fixed

Latch Timing

- Do the same type of thought experiment with latches
  - – Place one type of latch (if you have two) at the top of the sheet
  - – Let the distance from the top be the output settling time
  - – Roll the sheet into a cyclinder
- The difference is that the latches are not pinned
  - – As you roll the sheet, the latch position can move
  - – The circumference will be set by the longest cycle in the logic
    - Critical cycle might wrap the cylinder multiple times before closing
  - – Since the latches are not pinned
    - Clock skew has small effect on the overall machine timing

Hold Time Issues

- Latches don't solve the hold time problem
  - – Unless you can independently adjust all 4 edges of the clock
  - – If you use a single clock and use positive and negative edge
    - Skew will cause latches to have overlapping edges



    - Min delay must be larger than skew

Clock Skew Analysis

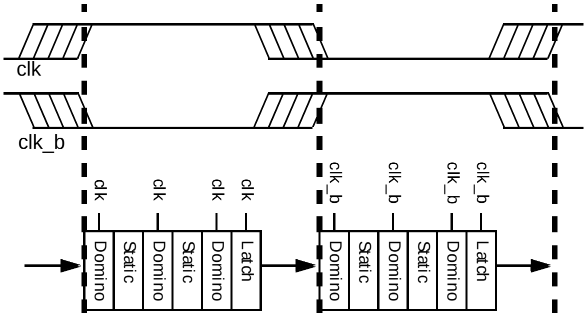
- Most simulation decks don’t give you skew numbers
  - Skew depends on “matching” between paths
- Part of the clocking system handed down gives skew values
  - There are often two different values given
    - Assumptions for long path, assumptions for short path
- Skew depend on the path difference to common ancestor:
  - Path between sequential element driven by clocks from the same local buffer
  - Path between sequential element driven by clocks from the different local buffer but same regional buffer
  - Path between sequential element driven by clocks from the different regional buffer
- For functionality issues, assume 30% path mismatch

What About Dynamic Logic

- Domino circuits use clocks to start evaluation
  - And also use clocks to latch results before they precharge
- Both of these timing edges are “hard”
  - They truly wait for the clock
  - Traditional domino circuits are large timing overheads
    - Skew budget, no time borrowing, latch delay
- Look at several ways to reduce this overhead
  - Remove hard edge from the latch
  - Remove all hard edges
- The cost of this technique is more complex clocking
  - System ends up looking similar to self-timed design
    - Completely data driven

Domino from a System Perspective

- Domino doesn't look so attractive in the context of a traditional pipeline



Legend:
Domino: One inverting dynamic gate
Static: One inverting static gate
Latch: One inverting tristate latch

1. Pay clock skew twice each phase
2. Balancing short phases is hard since there is no time borrowing
3. Latches become a significant fraction of the cycle time

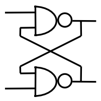
Hard Edges

- There are two hard edges we would like to remove:
- Eval clock
  - – The inputs must settle before the rising edge of eval clock
  - – But the gates don't evaluate until the eval clock
- Latch clock
  - – The outputs must settle before the falling edge of latch clock
    - While the data does flow through if it arrives early, the next stage is waiting for its evaluation clock, so this early arrival does not help
  - – Worse is the hold-time problem
    - Must not precharge the input to latch BEFORE the latch clock falls

- It turn out, you can remove all of these clock problems

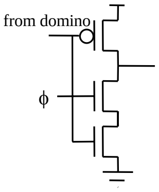
Latch Clock

- Since the logic is clocked, you don’t really need to clock latch
  - If the domino logic is dual rail
    - Have two outputs that are guaranteed to go to 1 in precharge
    - The outputs are Gate outputs are already _q1
    - Replace inverters after dynamic stage
      - With an simple SR latch



SR Latch

  - If the logic is not dual rail,
    - Still can remove edge
    - Build a partial tristate latch
    - Don’t gate the pullup with clock
    - If the precharge gate ever falls
      - Output will go high, independent of clock
    - But this combination has very bad noise margins!



TSPC Latch

Eval Clock

- If input data is not monotonic
  - Game is over, you need to wait for the clock
    - Since you need something to tell you inputs are valid
    - Can’t tolerate a 1->0 transition on your input
- If all the inputs are monotonic
  - Then they will indicate when they are valid
  - There is no need for a clock to gate the evaluation
    - Well need a clock to ensure precharge can happen,
    - But should not be on the critical path
      - Arrive before the data is valid
- For monotonic inputs
  - The gate will wait and fire when the data arrives

Eliminating Both Clocks

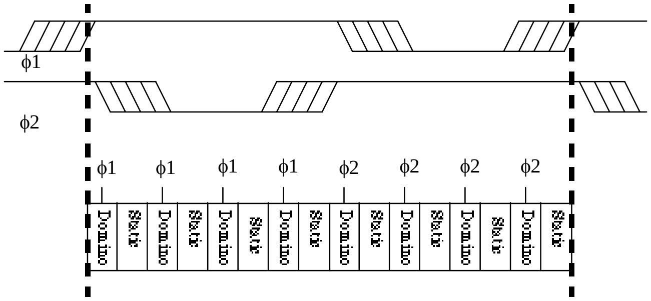
- What we want to do is create a long domino chain
  - But it would be nice to be able to create a domino loop
  - Just domino gates, with no latches, or clocks in the data flow
- How can this be done?
  - What would happen if
    - You skewed the clock slightly for each gate in the domino chain
    - The clock skew was less than the gate's delay
    - The clock skew was more than ½ the gate's delay
  - Gates in the chain would wait for their data
  - End of the chain would be a valid input to first gate
    - This gate would begin evaluation when first gate was in precharge
    - Would not precharge until some time after first gate was in eval

Building Many Clocks Is Painful

- Need to create and distribute all these clocks
  - So don't do that
- Don't need that many clocks
  - Domino chains work fine with a single clock
  - Need multiple clocks to form loops
- What is the minimum number of clocks needed?
  - Two, if you are willing to have hold time issues
  - Three without hold time
  - But we will look at 2 and 4 since these are symmetric
    - 4 is really two clocks and their complements

Skew-Tolerant Domino Circuits

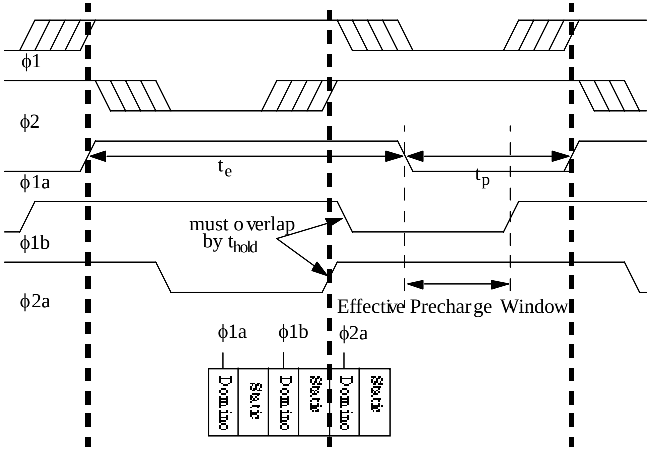
- How much clock skew could we tolerate given N clock phases?
  - Divide logic into N phases of T/N duration each.
  - Overlapping clocks eliminates need for latches
  - Extra overlap accommodates clock skew and time borrowing



- As with other domino techniques, budget skew on the transition from static to domino

Skew Tolerance Definitions

- Let’s call the cycle time, $T = t_e + t_p$ (evaluation + prechage)
- $t_p = t_{prech} + t_{skew}$; $t_e = T/N + t_{skew} + t_{hold}$
- Hence $t_{skew\text{-}max} = [T(N\text{-}1)/N - t_{prech} - t_{hold}] / 2$



Numerical Example

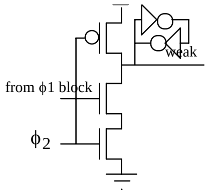
- Let t_{prech} = 4, long enough to:
  - – Precharge domino gate
  - – Make subsequent skewed static fall below V_t
- Assume t_{hold} is slightly negative for reasonable cell libraries
  - – Next phase can evaluate before precharge
    - Remember that precharge must ripple through static gate
  - – Conservatively bound t_{hold} at 0

N	t_{skew}	t_p
2	2	6
3	3.33	7.33
4	4	8
6	4.66	8.66
8	5	9

- Sweet spots:
  - – N=2 (fewest clocks), N=4 (good tolerance, 50% duty cycle)

Other Design Issues

- State is no longer stored in the latch at the end of a phase
  - – Instead, it is held by the first domino gate in the phase
  - – Use a “full keeper” to allow stop-clock operation



- All systems with overlapping clocks require min-delay checks
  - – The two phase system will have severe min-delay issues
  - – 4-phase has effectively no min-delay risk
    - Overlap of all four phases is at most very small
    - A minimum of 8 gates are in the cycle anyway

Summary

- Hard edges cost in performance because
  - Logic delay is not equally balanced initially
  - Chip variations, including clock skew changes the actual balance
  - The performance loss is do to the actual mismatch on chip
- Soft edges remove these performance overheads
  - Since timing variations can flow through clocked elements
  - Can be done for both dynamic logic and latches
  - But these systems often have worse hold-time issues
    - And these needed to be tested for worst-case conditions
- These ideas are not new (used in processors for a long time)
  - But rarely published