ECE 3401 Lecture 17

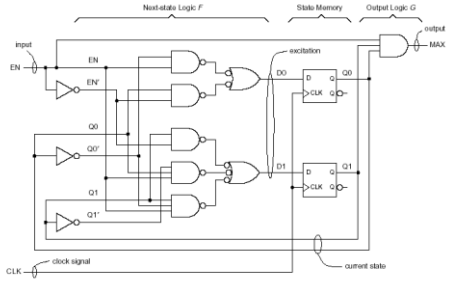
Memory & Timing Issues

Microprogramming (I)

Overview

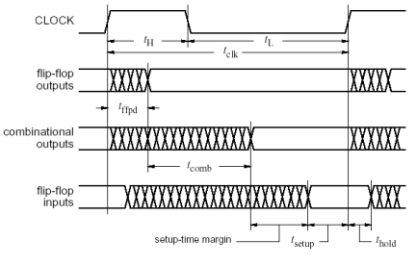
- Memories
  - Memory categories and timing
  - RAM:
    - SRAM
    - DRAM
- Timing issues
  - Sequential system timing requirements
  - Clock skew and clock jitter
  - Clock distribution

Clocked Synchronous State Machine Example



Setup and hold times requirements for state flip-flops must be satisfied.

Clocked Synchronous State Machine Timing



Timing margin equation: $t_{clk} \geq t_{ffpd} + t_{comb} + t_{setup}$

Setup time margin $= t_{clk} - t_{ffpd} - t_{comb} - t_{setup}$

Satisfying Timing Requirements

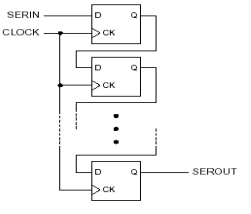
- The setup time margin can be made positive by making t_{clk} large enough.
  - Simple solution: slow down the system.
- Hold time requirement is independent of system clock.
  - Guarantee that minimum combinational logic delay is larger than hold time:

$$t_{ffpd} + t_{comb} \geq t_{hold}$$

Note that manufacturer's minimum delay specifications are needed.

Hold Time

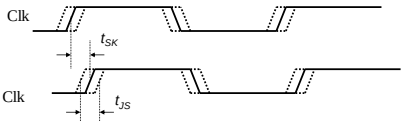
- To guarantee that the shift-register works, hold time t_{hold} must be smaller than flip-flop propagation delay t_{ffpd}.



Clock Nonidealities

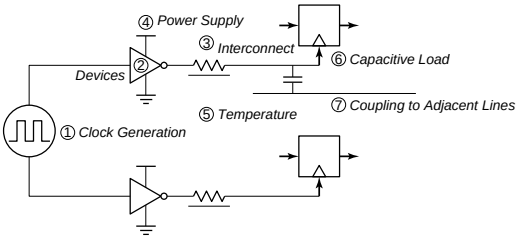
- Clock Skew
  - Spatial variations in equivalent clock edges
  - Mostly deterministic
- Clock Jitter
  - Temporal variations in consecutive clock edges
  - Mostly random
- Pulse Width Variation

Clock Skew and Jitter



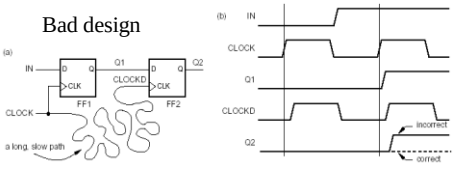
- Skew and jitter can affect the cycle times
- Clock skew can cause race conditions

Clock Uncertainties



Clock Skew

All flip-flops in clocked synchronous state machine should be clocked at the "same" time. Violating this rule may result in hold time violations.

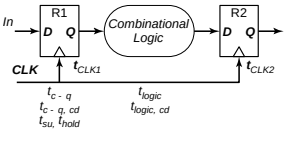


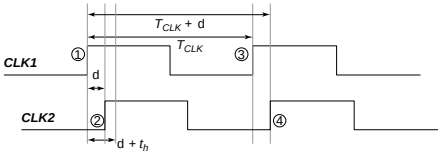
"Same" time means that difference between active edges should be small compared to hold time.

Clock rise and fall times should be short, in case flip-flops respond to different voltage levels. (Use similar flip-flops when possible.)

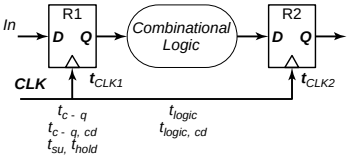
Clock Skew

- Positive clock skew
- Minimum clock cycle
  $T+\delta \geq t_{c\text{-}q} + t_{logic} + t_{su}$
- Positive δ *seems to improve performance*





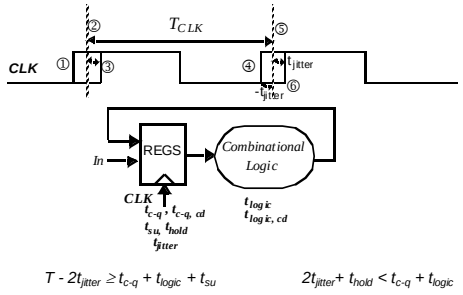
Clock Skew



Increasing skew will cause potential race conditions, violating:

$$\delta + t_{hold} < t_{c\text{-}q} + t_{logic}$$

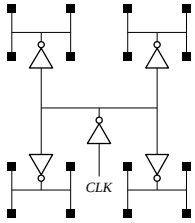
Clock Jitter



$T - 2t_{jitter} \geq t_{c\text{-}q} + t_{logic} + t_{su}$ $\quad 2t_{jitter} + t_{hold} < t_{c\text{-}q} + t_{logic}$

Clock Distribution

- Distribute clock in a tree fashion
- H-Tree

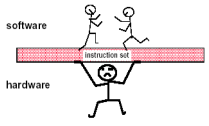


Microprogramming Overview

- Part 1 – Datapaths
  - Introduction
  - Datapath Example
    - Arithmetic Logic Unit (ALU)
    - Shifter
  - Datapath Representation and Control Word
- Part 2 – A Simple Computer
  - Instruction Set Architecture (ISA)
  - Single-Cycle Hardwired Control
- Part 3 – Multiple Cycle Hardwired Control
  - Single Cycle Computer Issues
  - Sequential Control Design

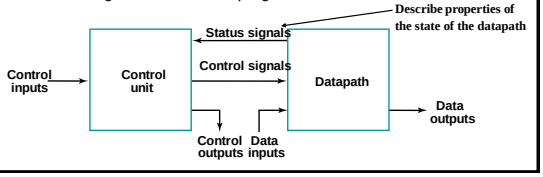
Introduction

- Computer Specification
  - *Instruction Set Architecture (ISA)* - the specification of a computer's appearance to a programmer at its lowest level
  - *Computer Architecture* - a high-level description of the hardware implementing the computer derived from the ISA
  - The architecture usually includes additional specifications such as speed, cost, and reliability.



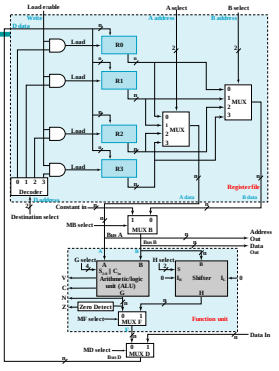
Introduction (Contd.)

- Simple computer architecture decomposed into:
  - Datapath: performing operations
    - A set of registers
    - Microoperations performed on the data stored in the registers
    - A control interface
  - Control unit: controlling datapath operations
    - Programmable & Non-programmable



Datapath Example

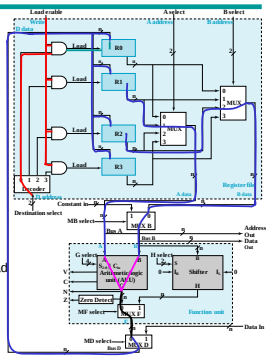
- Register file:
  - Four parallel-load regs
  - Two mux-based register selectors
  - Register destination decoder
- Microoperation Implementation
  - Mux B for external constant input
  - Buses A and B with external address and data outputs
  - Function Unit:
    - ALU and Shifter with Mux F for output select
    - Mux D for external data input
    - Logic for generating status bits V, C, N, Z



Datapath Example: Performing a Microoperation

Microoperation: R0 ← R1 + R2

- Apply 01 to A select to place contents of R1 onto Bus A
- Apply 10 to B select to place contents of R2 onto B data and apply 0 to MB select to place B data on Bus B
- Apply 0010 to G select to perform addition G = Bus A + Bus B
- Apply 0 to MF select and 0 to MD select to place the value of G onto BUS D
- Apply 00 to Destination select to enable the Load input to R0
- Apply 1 to Load Enable to force the Load input to R0 to 1 so that R0 is loaded on the clock pulse (not shown)
- The overall microoperation requires 1 clock cycle



Datapath Example: Key Control Actions for Microoperation Alternatives

Various microoperations:

- Perform a shift microoperation: apply 1 to MF select
- Use a constant in a micro-operation using Bus B: apply 1 to MB select
- Provide an address and data for a memory or output write microoperation – apply 0 to Load enable to prevent register loading
- Provide an address and obtain data for a memory or output read microoperation – apply 1 to MD select
- For some of the above, other control signals become don't cares