Sequential Logic

- Combinational logic:
  - Compute a function all at one time
  - Fast/expensive
  - e.g. combinational multiplier
- Sequential logic:
  - Compute a function in a series of steps
  - Slower/more efficient
  - e.g. shift/add multiplier
- Key to sequential logic: circuit has feedback
  - Use the result of one step as an input to the next

Circuits with feedback

- How to control feedback?
  - What stops values from cycling around endlessly?
  - This is an asynchronous sequential circuit

Simplest circuits with feedback: latch

- Two inverters form a static memory cell
  - Will hold value as long as it has power applied
    ![Latch Diagram]
  - How to get a new value into the memory cell?
    - Selectively break feedback path
    - Load new value into cell

Let's Use This Latch

- What happens?
  ![Latch Diagram]

What We Need:

- When inputs change...
  - Wait until combinational logic has finished and result is stable...
  - Then sample the output value and save...
- Feedback from the combinational logic
  - Key idea: we sample the result at the right time, i.e. when it is ready
    - Only then do we update the stored value
- How do we know when to sample?
- How do we know when the inputs changed?
- How do we know how long to wait?

What We Need:

- A circuit that can sample a value
- A signal that says when to sample
- Edge-triggered D flip-flop (register)
  - Sample on positive edge of clock
  - Holds value until next positive edge
  - Most common storage element
- Clock
  - Periodic signal, each rising edge signals D flip-flops to sample
  - All registers sample at the same time
Let's Use This D flip-flop

- Does this work?
- What do we need to say about the inputs X₁, X₂, ...?
- This is a synchronous sequential circuit

Example - Circuit with Feedback

- Output is a function of arbitrarily many past inputs

Example - Circuit without Feedback

- Output is a function of the input sampled at three different points in time

Registers

- Sample data using clock
- Hold data between clock cycles
- Computation (and delay) occurs between registers

Examples (cont’d)

- Output is a function of inputs and the previous state of the circuit

Timing Methodologies (cont’d)

- Definition of terms
  - setup time: minimum time before the clocking event by which the input must be stable (T_s)
  - hold time: minimum time after the clocking event until which the input must remain stable (T_h)

Example 5: Circuit with Feedback

Example 6: Circuit without Feedback

Example 7: Timing Methodologies (cont’d)
Sequential Logic

Typical timing specifications
- Positive edge-triggered D flip-flop
- setup and hold times
- minimum clock width
- propagation delays (low to high, high to low, max and typical)

![Typical Timing Specifications Diagram]

System Clock Frequency
- Register transfer must fit into one clock cycle
  - reg t_{sw} < CL t_{pu} < reg t_{px} < T_{clk}
  - Use maximum delays
  - Find the "critical path"
  - Longest register-register delay

![System Clock Frequency Diagram]

Shift register
- Holds samples of input
  - store last 4 input values in sequence
- 4-bit shift register

![Shift Register Diagram]

4-bit Universal shift register
- Holds 4 values
  - serial or parallel inputs
  - serial or parallel outputs
  - permits shift left or right
  - shift in new values from left or right

![4-bit Universal Shift Register Diagram]

Design of Universal Shift Register
- Consider one of the four flip-flops
  - new value at next clock cycle:
    - clear s0 s1 new value
    - 1 1 0
    - 0 1 output value of FF to left (shift right)
    - 0 0 1 output value of FF to right (shift left)
    - 0 1 1 input

![Design of Universal Shift Register Diagram]
**Binary counter**

- Next state function for bit $i$
  - XOR acts like a "programmable" inverter
  - if bits 0-i are 1, then toggle bit $i$
  - requires an i-input AND for bit $i$
- Synchronous: outputs all change when clock ticks

**Other Counters: cheaper/faster**

- Sequences through a fixed set of patterns
  - in this case, 1000, 0010, 0001, 0001
  - if one of the patterns is its initial state (by loading or set/reset)

- Mobius (or Johnson) counter
  - in this case, 1000, 1100, 1110, 1111, 0011, 0001, 0000