CS 2630
Computer Organization
Meeting 14: Sequential logic
Brandon Myers
University of Iowa
Ingredients for a processor

• Combinational logic
  • arithmetic and bitwise operations
  • multiplexors (pick 1 out of N inputs)
  • control logic
  • branch calculation
  • address calculation

• Synchronous logic
  • registers
  • program counter, or PC
  • data memory
  • instruction memory
Basic memory element: flip-flop

- on the rising edge of the clock signal, the value on input D is "captured" into the flip-flop

- value of the output Q is the value stored in the flip-flop
Clock signal

- Analogy with a clock on the wall:
  - every *tick* happens “instantaneously” at the *rising edge*
  - the *period* is the time between ticks
- period of a wall clock: 1 second
- period of a typical digital logic clock: nanoseconds or less
Reading waveforms

- at the rising edge, sample D’s current value
- some small delay later, Q becomes that value
Draw the rest of signal Q. What is the value of Q at (a) (b) and (c)? (enter three numbers {0,1}).

Space available for you to draw D, as well.
Administrivia

• Midterm grades are up; pass back near end of class
• Reminder:
  • HW3 is due on Friday, 11:59pm
Register: collection of N flip-flops to store N bits

- Example: 2-bit register
Building an accumulator

i.e., a circuit that would add a sequence of numbers, like this code

```java
int[] A = {5, 1, 3, 1};
int sum = 0;
for (int t=0; t<N; t++) {
    sum += A_t;
}
```

first attempt:

```
A_t
   +
   ---------------
   |                  |
   | 32-bit sum      |
   |                  |
   | sum             |
```

problems

- how do we know when to change $A_t$ to $A_{t+1}$?
- what if some bits of sum change faster than others?
- how do we initialize sum to 0?
Building an accumulator

i.e., a circuit that would add a sequence of numbers, like this code

```c
int[] A = {5, 1, 3, 1};
int sum = 0;
for (int t=0; t<N; t++) {
    sum += A_t;
}
```

second attempt:

![Diagram of an accumulator circuit](image)
Peer instruction

Change the accumulator to have another input, Reset, which resets sum to 0 when it is 1.
Propagation delay in combinational logic: shown with waveforms

A  B

+    cout

Sum

new A=1 arrives

new B=1 arrives

Sum and cout stable here
Next steps

• learn about Finite State Machines, a way of thinking about circuits with flip-flops and combinational logic
• arrange stored bits into large memories and learn how to lookup values using an address