State Elements: Circuits That Remember

CPU, GPU, now PPU!⇒Ageia’s “PhysX” chip will accelerate physics common to video games: rigid, soft body & fluid dynamics, collision detection, finite element analysis, hair & clothing sim! www.ageia.com/technology.html

Review...

- ISA is very important abstraction layer
  - Contract between HW and SW
- Basic building blocks are logic gates
- Clocks control pulse of our circuits
- Voltages are analog, quantized to 0/1
- Circuit delays are fact of life
- Two types
  - Stateless Combinational Logic (&,|,~), in which output is function of input only
  - State circuits (e.g., registers)

Accumulator Example

Want: \[ S=0; \text{for } (i=0; i<n; i++) \]
\[ S = S + X_i \]

First try...Does this work?

Nope!
Reason #1... What is there to control the next iteration of the ‘for’ loop?
Reason #2... How do we say: ‘S=0’?

Second try...How about this? Yep!

Rough timing...
Register Details... What’s in it anyway?

- n instances of a “Flip-Flop”, called that because the output flips and flops between 0,1
- D is “data”
- Q is “output”
- Also called “d-q Flip-Flop”, “d-type Flip-Flop”

What’s the timing of a Flip-flop? (1/2)

- Edge-triggered d-type flip-flop
  - This one is “positive edge-triggered”
  - “On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored.”

What’s the timing of a Flip-flop? (2/2)

- Edge-triggered d-type flip-flop
  - This one is “positive edge-triggered”
  - “On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored.”

Administrivia - Midterm

- Your TAs and readers stayed up until 6:30am to get your exams back to you!
  - $\bar{x} = 47$, Median: 48, $\sigma$: 14.4

Accumulator Revisited (proper timing 1/2)

Accumulator Revisited (proper timing 2/2)
Piplining to improve performance (1/2)

Finite State Machines Introduction

Hardware Implementation of FSM

General Model for Synchronous Systems
Peer Instruction

A. HW feedback akin to SW recursion
B. We can implement a D-Q flipflop as simple CL (And, Or, Not gates)
C. You can build a FSM to signal when an equal number of 0s and 1s has appeared in the input.

```
ABC
1: FFP
2: FTF
3: FTF
4: FTT
5: TFF
6: TTF
7: TTT
8: TTT
```

“And In conclusion…”

• We use feedback to maintain state
• Register files used to build memories
• D-FlipFlops used to build Register files
• Clocks tell us when D-FlipFlops change
  • Setup and Hold times important
• We pipeline big-delay CL for faster clock
• Finite State Machines extremely useful
  • You’ll see them in HW classes (150,152) & 164