CS 61C: Great Ideas in Computer Architecture *Timing and State*

Instructors:
Krste Asanovic, Randy H. Katz
http://inst.eecs.Berkeley.edu/~cs61c/fa12

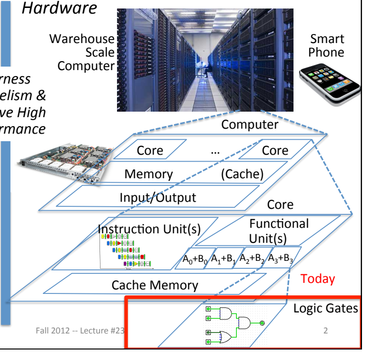
You are Here!

Software

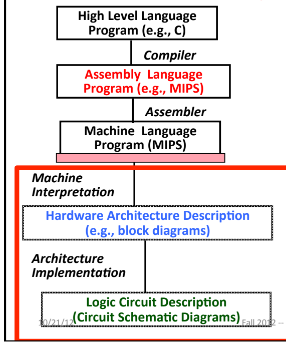
- Parallel Requests
  Assigned to computer
  e.g., Search "Katz"
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions
  All gates @ one time
- Programming Languages

Hardware

Harness Parallelism & Achieve High Performance



Levels of Representation/ Interpretation

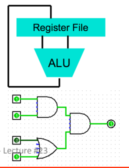

High Level Language Program (e.g., C)
Compiler
Assembly Language Program (e.g., MIPS)
Assembler
Machine Language Program (MIPS)
Machine Interpretation
Hardware Architecture Description (e.g., block diagrams)
Architecture Implementation
Logic Circuit Description (Circuit Schematic Diagrams)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw  $t0, 0($2)
lw  $t1, 4($2)
sw  $t1, 0($2)
sw  $t0, 4($2)
```

Anything can be represented as a *number*, i.e., data or instructions

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```



Review

- Real world voltages are analog, but are quantized to represent logic 0 and logic 1
- Transistors are just switches, combined to form gates: AND, OR, NOT, NAND, NOR
- Truth table can be mapped to gates for combinational logic design
- Boolean algebra allows minimization of gates

Agenda

- State Elements
- Finite State Machines
- And in Conclusion, …

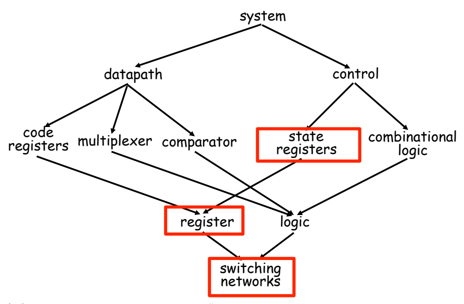
Agenda

- State Elements
- Finite State Machines
- And in Conclusion, …

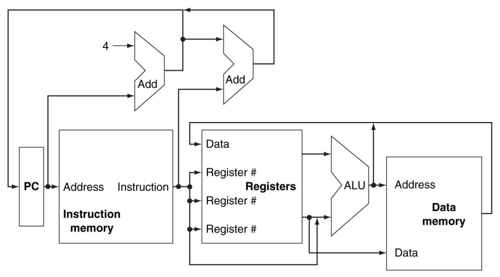
Type of Circuits

- *Synchronous Digital Systems* consist of two basic types of circuits:
  - Combinational Logic (CL) circuits
    - – Output is a function of the inputs only, not the history of its execution
    - – E.g., circuits to add A, B (ALUs)
    - – Last lecture was CL
  - Sequential Logic (SL)
    - Circuits that “remember” or store information
    - aka “State Elements”
    - E.g., memories and registers (Registers)
    - Today’s lecture is SL

Design Hierarchy



A Conceptual MIPS Datapath

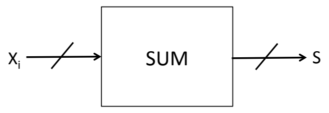


Uses for State Elements

- Place to store values for later re-use:
  - – Register files (like \$1-\$31 on the MIPS)
  - – Memory (caches, and main memory)
- *Help control flow of information between combinational logic blocks*
  - – State elements hold up the movement of information at input to combinational logic blocks to allow for orderly passage

Accumulator Example

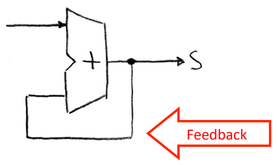
Why do we need to control the flow of information?



Want:
```
S=0;
for (i=0;i<n;i++)
   S = S + X_i
```

Assume:
- Each X value is applied in succession, one per cycle
- After n cycles the sum is present on S

First Try: Does this work?

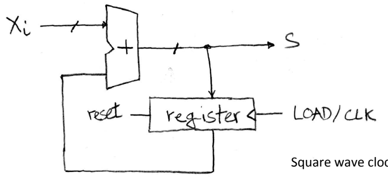


No!

Reason #1: How to control the next iteration of the ‘for’ loop?

Reason #2: How do we say: ‘S=0’?

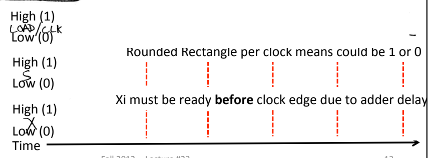
Second Try: How About This?



Register is used to hold up the transfer of data to adder

Square wave clock sets when things change

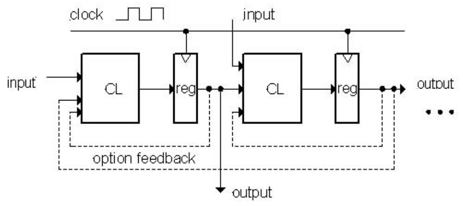
Rough timing …



Rounded Rectangle per clock means could be 1 or 0

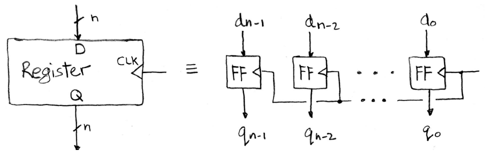
Xi must be ready **before** clock edge due to adder delay

Model for Synchronous Systems



- Collection of Combinational Logic blocks separated by registers
- Feedback is optional
- Clock signal(s) connects only to clock input of registers
- Clock (CLK): steady square wave that synchronizes the system
- Register: several bits of state that samples on rising edge of CLK (positive edge-triggered) or falling edge (negative edge-triggered)

Register Internals



- n instances of a "Flip-Flop"
- Flip-flop name because the output flips and flops between 0 and 1
- D is "data input", Q is "data output"
- Also called "D-type Flip-Flop"

Camera Analogy Timing Terms

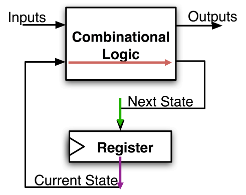
- Want to take a portrait – timing right before and after taking picture
- *Set up time* – don't move since about to take picture (open camera shutter)
- *Hold time* – need to hold still after shutter opens until camera shutter closes
- *Time click to data* – time from open shutter until can see image on output (viewfinder)

Hardware Timing Terms

- Setup Time: when the input must be stable *before* the edge of the CLK
- Hold Time: when the input must be stable *after* the edge of the CLK
- "CLK-to-Q" Delay: how long it takes the output to change, measured from the edge of the CLK

FSM Maximum Clock Frequency

- What is the maximum frequency of this circuit?

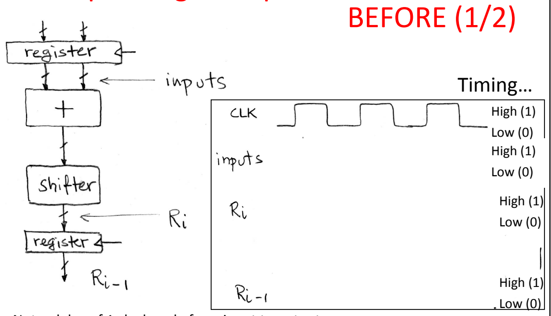


Hint:
Frequency = 1/Period

Max Delay = Setup Time + CLK-to-Q Delay + CL Delay

Pipelining to Improve Performance: BEFORE (1/2)

Timing…

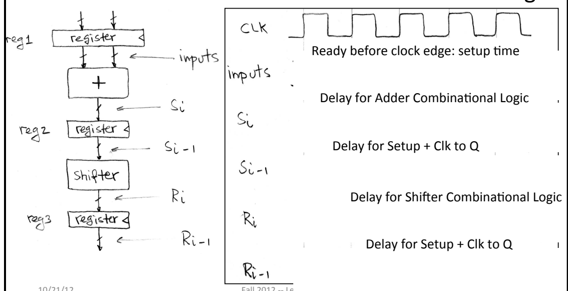


Note: delay of 1 clock cycle from input to output.
Clock period limited by propagation delay of adder/shifter

Pipelining to Improve Performance (2/2)

- Insertion of register allows higher clock frequency
- More outputs per second

Timing…

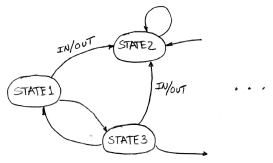

reg1
register
inputs
+
Si
reg2
register
Si-1
shifter
Ri
reg3
register
Ri-1
CLK
inputs
Ready before clock edge: setup time
Delay for Adder Combinational Logic
Si
Delay for Setup + Clk to Q
Si-1
Delay for Shifter Combinational Logic
Ri
Delay for Setup + Clk to Q
Ri-1

Agenda

- State Elements
- Finite State Machines
- And, in Conclusion, …

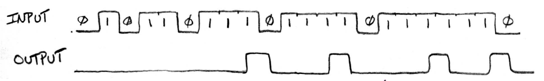
Another Great (Theory) Idea: Finite State Machines (FSM)

- You may have seen FSMs in other classes
- Same basic idea
- Function can be represented with a "state transition diagram"
- With combinational logic and registers, any FSM can be implemented in hardware

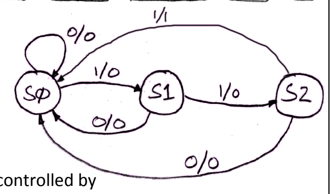


Example: 3 Ones FSM

FSM to detect the occurrence of 3 consecutive 1's in the Input



Draw the FSM …



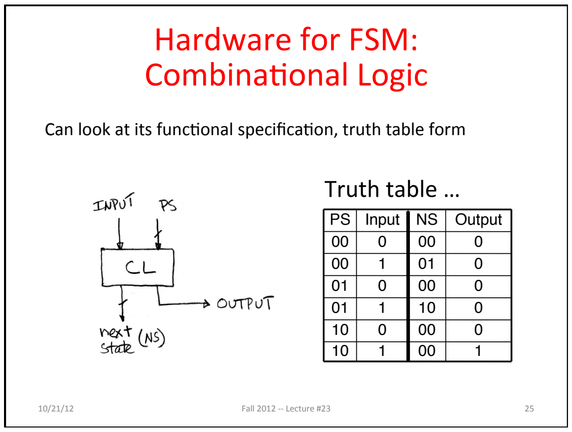
Assume state transitions are controlled by the clock: On each clock cycle the machine checks the inputs and moves to a new state and produces a new output …

Hardware Implementation of FSM

Register needed to hold a representation of the machine's state.
Unique bit pattern for each state.

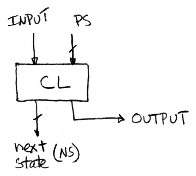
Combinational logic circuit is used to implement a function maps from *present state (PS)* and *input* to *next state (NS)* and *output*.

The register is used to break the feedback path between Next State (NS) and Prior State (PS), controlled by the clock



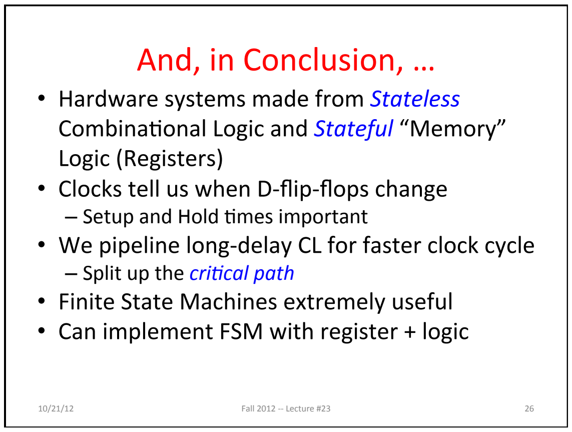
Hardware for FSM: Combinational Logic

Can look at its functional specification, truth table form



Truth table …

PS	Input	NS	Output
00	0	00	0
00	1	01	0
01	0	00	0
01	1	10	0
10	0	00	0
10	1	00	1



And, in Conclusion, …

- Hardware systems made from *Stateless* Combinational Logic and *Stateful* "Memory" Logic (Registers)
- Clocks tell us when D-flip-flops change
  - Setup and Hold times important
- We pipeline long-delay CL for faster clock cycle
  - Split up the *critical path*
- Finite State Machines extremely useful
- Can implement FSM with register + logic