EECS150 - Digital Design
Lecture 4 - Synchronous Digital Systems Review Part 2

January 27, 2011

John Wawrzynek
Electrical Engineering and Computer Sciences
University of California, Berkeley

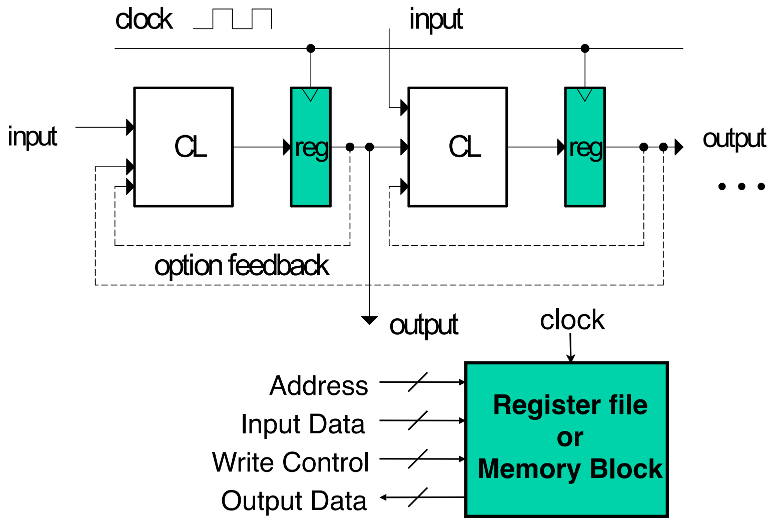
http://www-inst.eecs.berkeley.edu/~cs150

Outline

- Topics in the review, you have already seen in CS61C, and possibly EE40:
  1. Digital Signals.
  2. General model for synchronous systems.
  3. Combinational logic circuits
  4. Flip-flops, clocking

Only Two Types of Circuits Exist

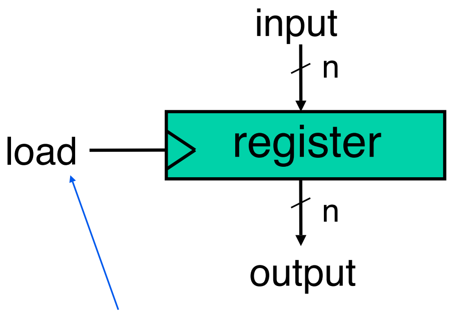
- Combinational Logic Blocks (CL)
- State Elements (registers)



- State elements are mixed in with CL blocks to control the flow of data.
- Sometimes used in large groups by themselves for "long-term" data storage.

State Elements: circuits that store info

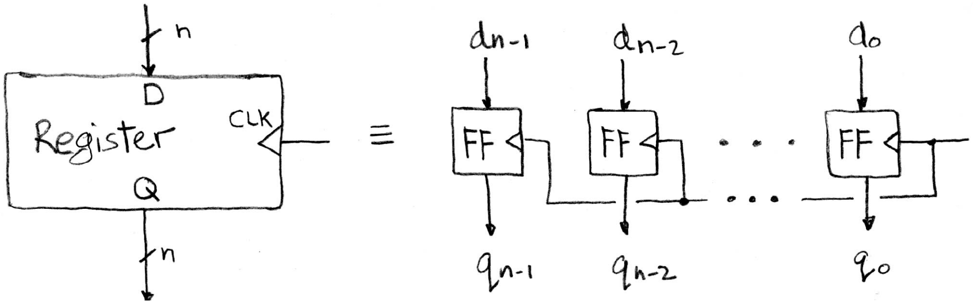
- Examples: registers, memories
- Register: Under the control of the "load" signal, the register captures the input value and stores it indefinitely.



often replace by clock signal (clk)

- The value stored by the register appears on the output (after a small delay).
- Until the next load, changes on the data input are ignored (unlike CL, where input changes change output).
- These get used for short term storage (ex: register file), and to help move data around the processor.

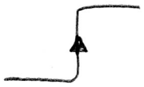
Register Details…What's inside?



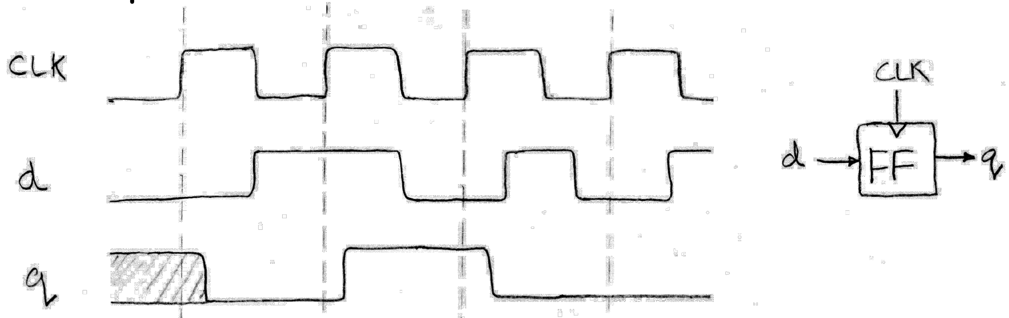
- n instances of a "Flip-Flop"
- Flip-flop name because the output flips and flops between and 0,1
- D is "data", Q is "output"
- Also called "d-type Flip-Flop"

Flip-flop Timing

- 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."
- Example waveforms:



Uses for State Elements

1) As a place to store values for some indeterminate amount of time:

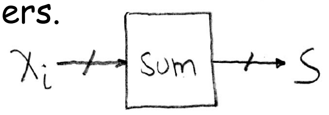
- Register files (like $1-$31 on the MIPS)
- Memory (caches, and main memory)

2) Help control the flow of information between combinational logic blocks.

- State elements are used to hold up the movement of information at the inputs to combinational logic blocks and allow for orderly passage.

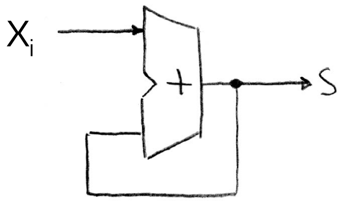
Accumulator Circuit Example

Assume X is a vector of N integers, presented to the input of our accumulator circuit one at a time (one per clock cycle), so that after N clock cycles, S hold the sum of all N numbers.



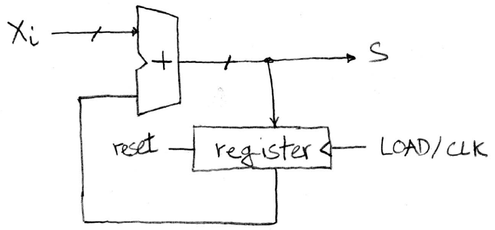
```
S=0; Repeat N times
        S = S + X;
```

- We need something like this:



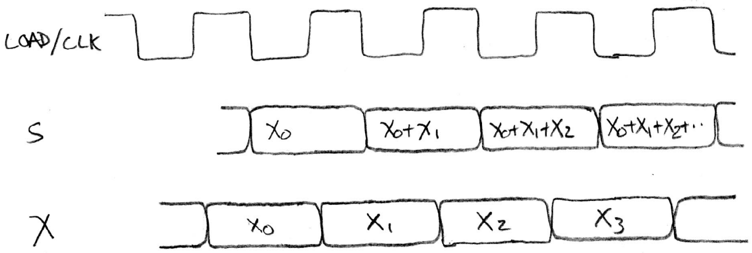
- But not quite.
- Need to use the clock signal to hold up the feedback to match up with the input signal.

Accumulator Circuit



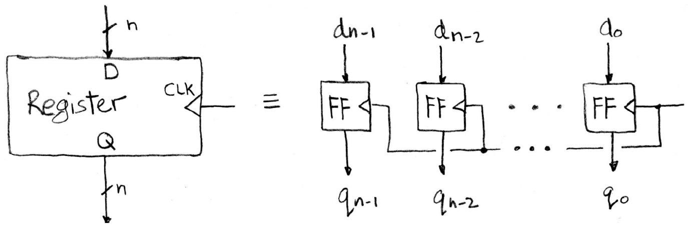
- Put register, with clock signal controlling its load, in feedback path.
- On each clock cycle the register prevents the new value from reaching the input to the adder prematurely. (The new value just waits at the input of the register).

Timing:

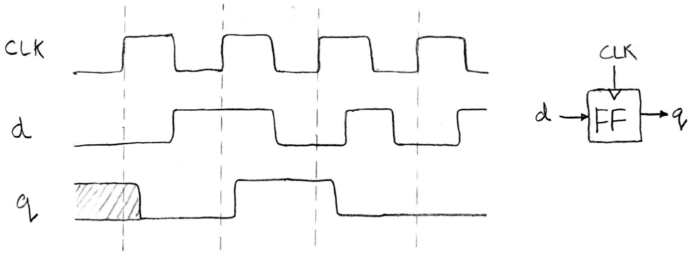


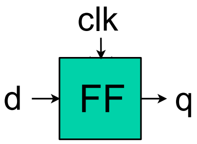
Register Details (again)

- A n-bit wide register is nothing but a set of flip-flops (1-bit wide registers) with a common load/clk signal.

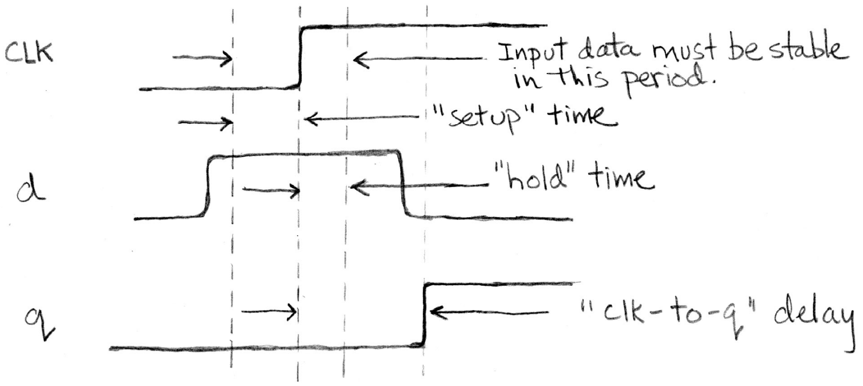


- A flip-flop captures its input on the edge of the clock (rising edge in this case - positive edge flip-flop). The new input appears at the output after a short delay.





Flip-Flop Timing Details



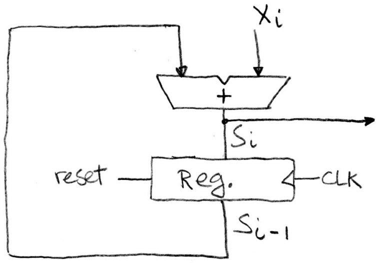
Three important times associated with flip-flops:

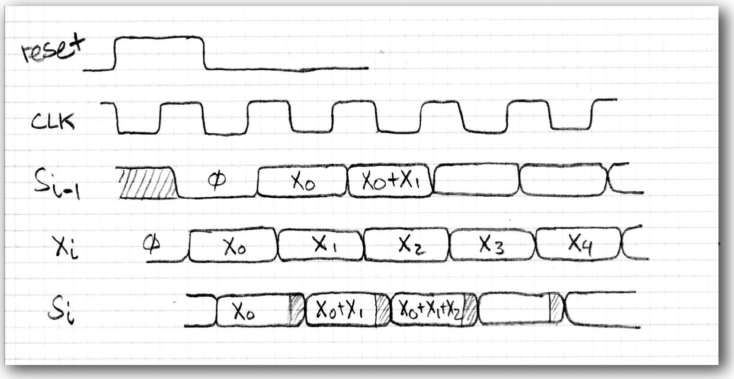
setup time

hold time

clock-to-q delay.

Accumulator Revisited

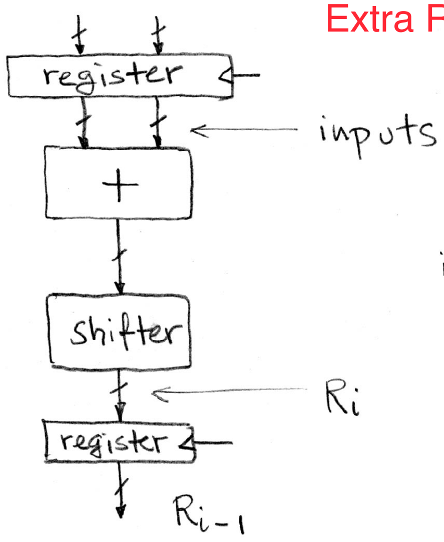




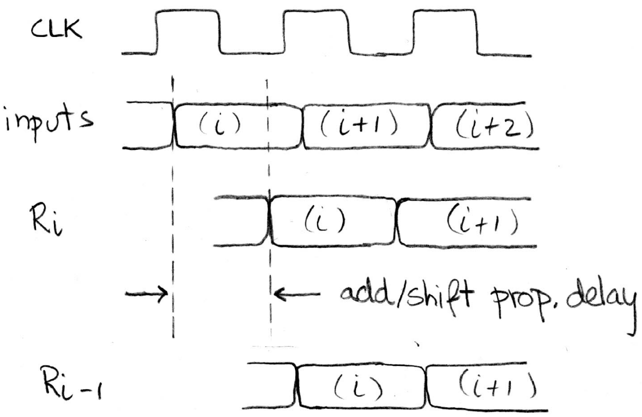
- Note:
  - Reset signal (synchronous)
  - Timing of X signal is not known without investigating the circuit that supplies X. Here we assume it comes just after S_{i-1}.
  - Observe transient behavior of S_i.

Pipelining to improve performance (1/2)

Extra Register are often added to help speed up the clock rate.



Timing...

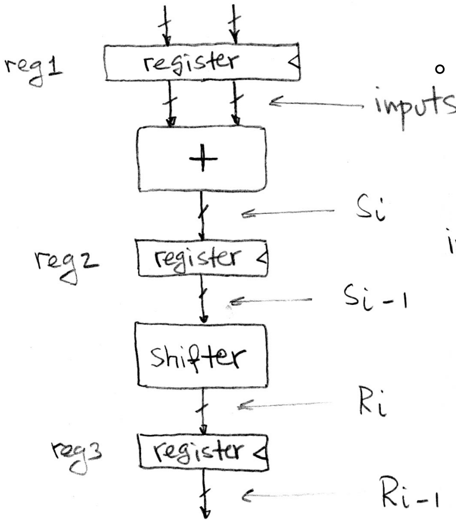


Note: delay of 1 clock cycle from input to output.

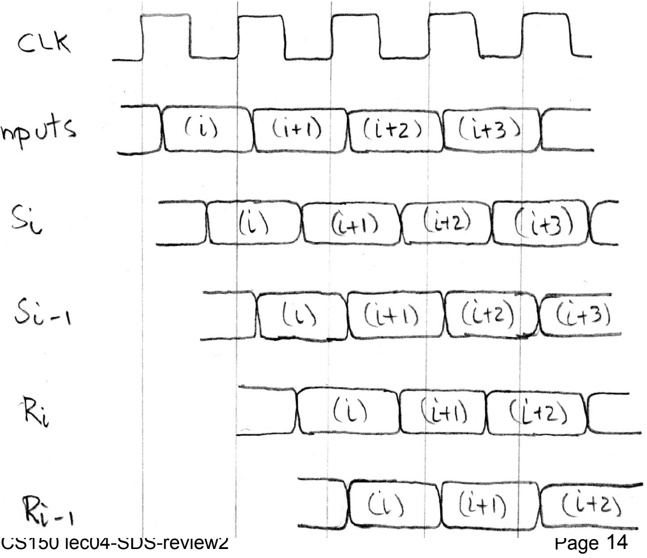
Clock period limited by propagation delay of adder/shifter.

Pipelining to improve performance (2/2)

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

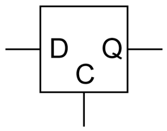


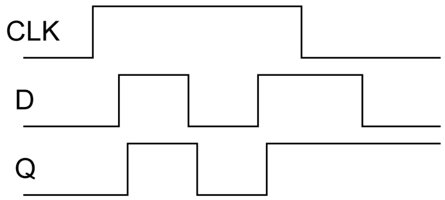
Timing...



Level-sensitive Latch

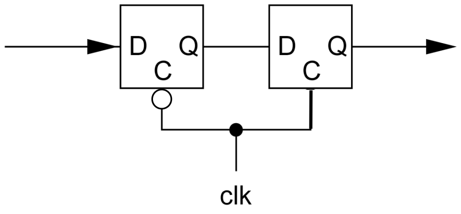
Positive Level-sensitive latch:





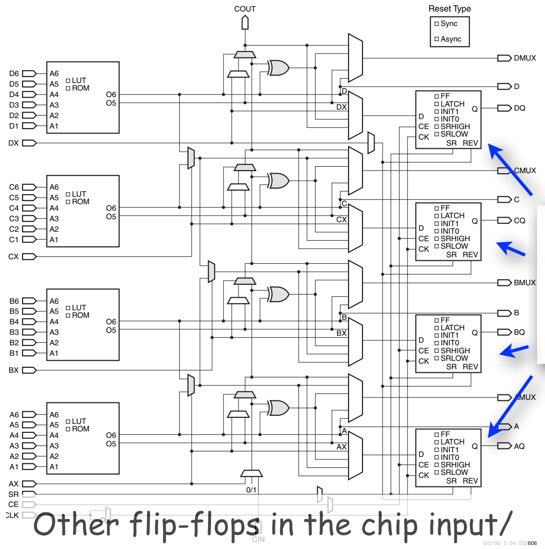
When CLK is high, latch is transparent, when clk is low, latch retains previous value.

Positive Edge-triggered flip-flop built from two level-sensitive latches:

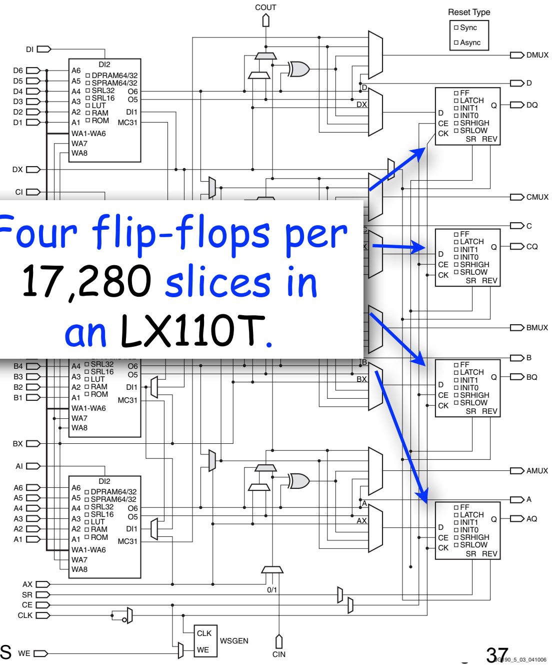


Flip-flops on Virtex5 FPGA

SLICEL



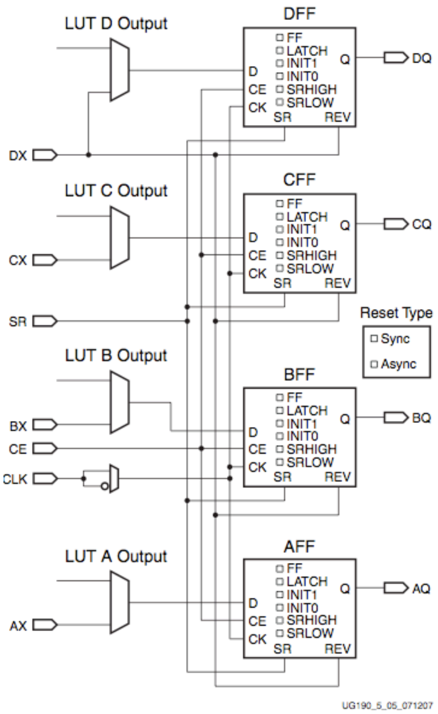
SLICEM



Four flip-flops per 17,280 slices in an LX110T.

Other flip-flops in the chip input/output cells, and in the form of registers in the DSP slices and memory block interfaces.

Virtex5 Slice Flip-flops

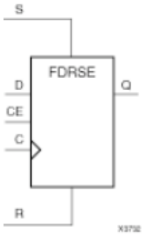


4 flip-flops / slice (corresponding to the 4 6-LUTs)

Each takes input from LUT output or primary slice input.

Edge-triggered FF vs. level-sensitive latch.
Clock-enable input (can be set to 1 to disable) (shared).
Positive versus negative clock-edge.
Synchronous vs. asynchronous reset.
SRHIGH/SRLOW select reset (SR) set.
REV forces opposite state.
INIT0/INIT1 used for global reset (not shown - usually just after power-on and configuration).

Virtex5 Flip-flops “Primitives”

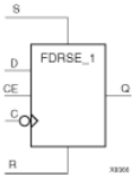


D Flip-Flop with Synchronous Reset and Set and Clock Enable

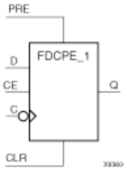
Provided by the CAD tools. This maps to single slice flip-flop.

Logic Table

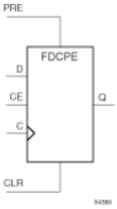
Inputs					Output
R	S	CE	D	C	Q
1	-	-	-	↑	0
0	1	-	-	↑	1
0	0	0	-	-	No Change
0	0	1	1	↑	1
0	0	1	0	↑	0



Negative-Clock Edge, Synchronous Reset and Set, and Clock Enable



Negative-Edge Clock, Clock Enable, and Asynchronous Preset and Clear



Clock Enable and Asynchronous Preset and Clear