CME 341: Synchronous Logic, Part 2

Brian Berscheid

Department of Electrical and Computer Engineering
University of Saskatchewan

Today's agenda

1. Basic features of flip-flops

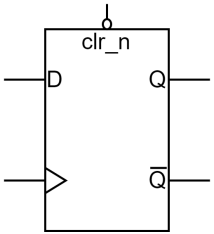
Basic features of flip-flops

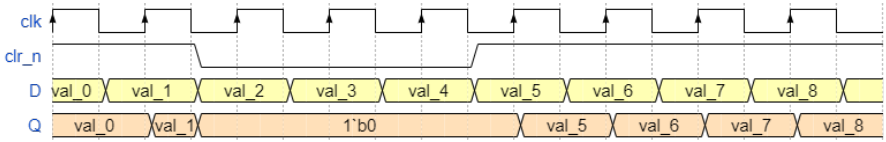
Flip-flops

- Flip-flops (FFs) are the key element of synchronous logic
- There are many different types of FFs; the most commonly used type in FPGAs is the D-FF (introduced last time)
- Focusing on D-FFs, there are a number of sub-types which have various input-output and timing relationships
- This section introduces the D-FF sub-types and investigates their behavioral Verilog descriptions
  - ▷ Clear/Reset functionality (asynchronous or synchronous)
  - ▷ Set/Preset functionality (asynchronous or synchronous)
  - ▷ Data load functionality (asynchronous or synchronous)

D-FF with asynchronous clear

Example shows active low, but could also be active high.





"When a negative edge of the clear signal occurs, immediately reset Q to 0.
If a clock edge occurs while clear is active, hold Q at 0.
Otherwise, set Q to D when the clock arrives."

D-FF with asynchronous clear - Verilog description

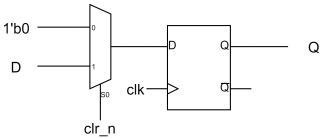
Translating description of operation into code

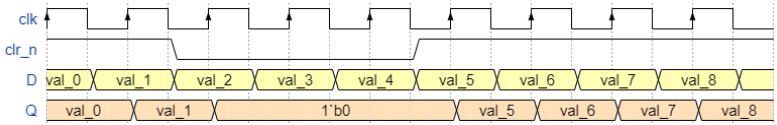
"When a negative edge of the clear signal occurs, immediately reset Q to 0.
If a clock edge occurs while clear is active, hold Q at 0.
Otherwise, set Q to D when the clock arrives."

```
reg q;

always @ (posedge clk or negedge clr_n)
  if (clr_n == 1'b0)
    q = 1'b0;
  else
    q = d;
```

D-FF with synchronous clear

Example shows active low, but could also be active high.





"When a positive edge of the clock signal occurs, check if clear is active. If so, set Q to 0. Otherwise, set Q to D."

D-FF with synchronous clear - Verilog description

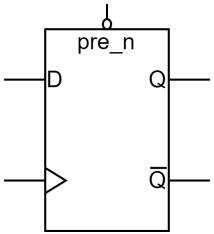
Translating description of operation into code

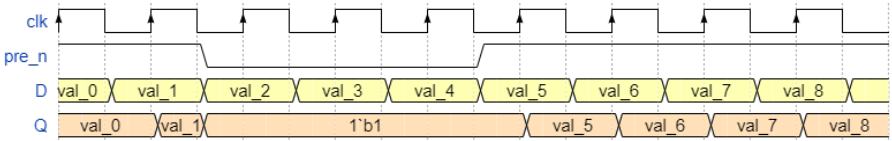
"When a positive edge of the clock signal occurs, check if clear is active. If so, set Q to 0. Otherwise, set Q to D."

```
reg q;

always @ (posedge clk)
  if (clr_n == 1'b0)
    q = 1'b0;
  else
    q = d;
```

D-FF with asynchronous preset

Similar to clear, but sets FF output to 1





"When a negative edge of the preset signal occurs, immediately set Q to 1.
If a clock edge occurs while preset is active, hold Q at 1.
Otherwise, set Q to D when the clock arrives."

D-FF with asynchronous preset - Verilog description

Translating description of operation into code

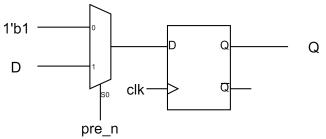
"When a negative edge of the preset signal occurs, immediately set Q to 1.
If a clock edge occurs while preset is active, hold Q at 1.
Otherwise, set Q to D when the clock arrives."

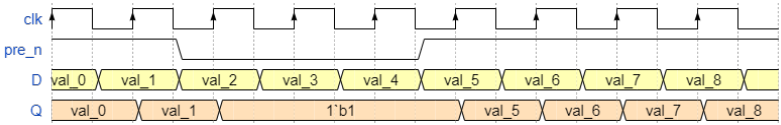
```
reg q;

always @ (posedge clk or negedge pre_n)
  if (pre_n == 1'b0)
    q = 1'b1;
  else
    q = d;
```

Try to sketch the timing diagram and hardware for a D-FF with synchronous preset. Then write the corresponding Verilog code.

D-FF with synchronous preset

Example shows active low, but could also be active high.





"When a positive edge of the clock signal occurs, check if preset is active. If so, set Q to 1. Otherwise, set Q to D."

D-FF with synchronous preset - Verilog description

"When a positive edge of the clock signal occurs, check if preset is active. If so, set Q to 1. Otherwise, set Q to D."

```
reg q;

always @ (posedge clk)
  if (pre_n == 1'b0)
    q = 1'b1;
  else
    q = d;
```

Key observations

- When an asynchronous clear or preset is used, the corresponding signal must be added to the sensitivity list with a 'posedge' or 'negedge' designation.
  - ▷ Must be accompanied by a test in the procedure body:

    Test for signal $==$ 1 if posedge
    Test for signal $==$ 0 if negedge

- The FF primitive used in CME 341 (DFFE) includes asynchronous active low preset and clear ports

- Synchronous clear and preset are achieved by placing additional logic in front of the FF to control its D input

Example: D-FF with asynchronous preset and asynchronous clear

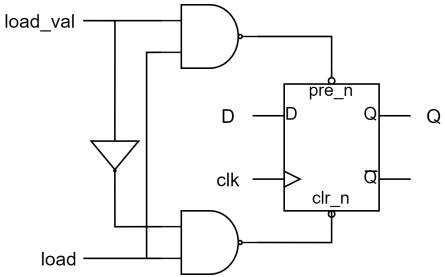
```
reg q;
always @ (posedge clk or negedge pre_n or negedge clr_n)
  if (clr_n == 1'b0)     // clear overrides set in this example
    q = 1'b0;
  else if (pre_n == 1'b0)
    q = 1'b1;
  else
    q = d;
```

Note: Just an example of the behavior description process...
It is not usually necessary to have both asynchronous preset and clear.

Asynchronous load

Load an input signal rather than a constant

```
reg q;

always @ (posedge clk or posedge load)
  if (load == 1'b1)
    q = load_val;  // load_val is a variable
  else
    q = d;
```



Challenge:
Write a Verilog module which implements an 8-bit counter that is asynchronously preloadable and has a count enable.

Solution in video...

A FF with two clocks?

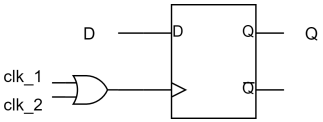
- Suppose you write the following Verilog code:

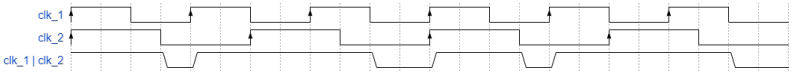
```
reg q;

always @ (posedge clk_1 or posedge clk_2)
  q = d;
```

- Two posedges in sensitivity list, neither being used as async preset/clear
- Intention is to copy d into q whenever a posedge of either clk_1 or clk_2 occurs
- What hardware is built?

A FF with two clocks - common misunderstanding

Can't OR the two clocks together... clock edges will be missed





This circuit does not work as intended.
It is not possible to build a FF with two clocks using FPGA technology!

Thank you!
Have a great day!