Digital Design: An Embedded Systems Approach Using Verilog

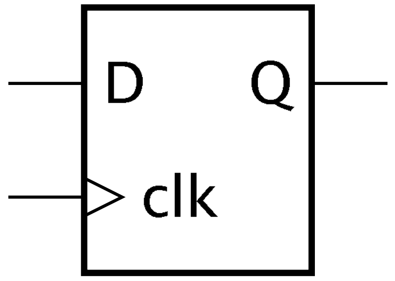
Chapter 4
Sequential Basics

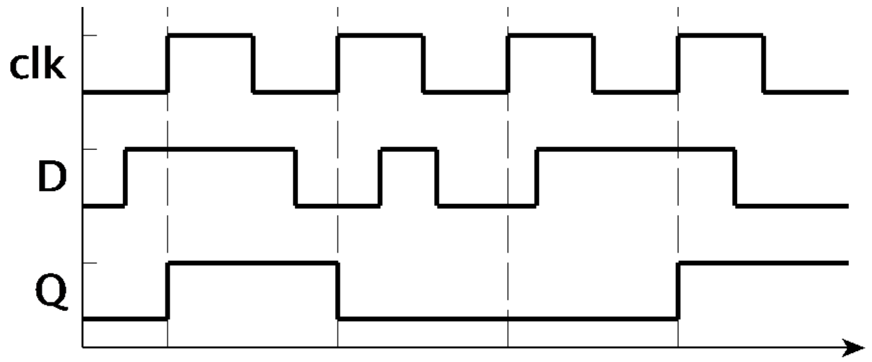
Sequential Basics

- Sequential circuits
  - Outputs depend on current inputs and previous inputs
  - Store *state*: an abstraction of the history of inputs
- Usually governed by a periodic clock signal

D-Flipflops

- 1-bit storage element
  - We will treat it as a basic component





- Other kinds of flipflops
  - SR (set/reset), JK, T (toggle)

Registers

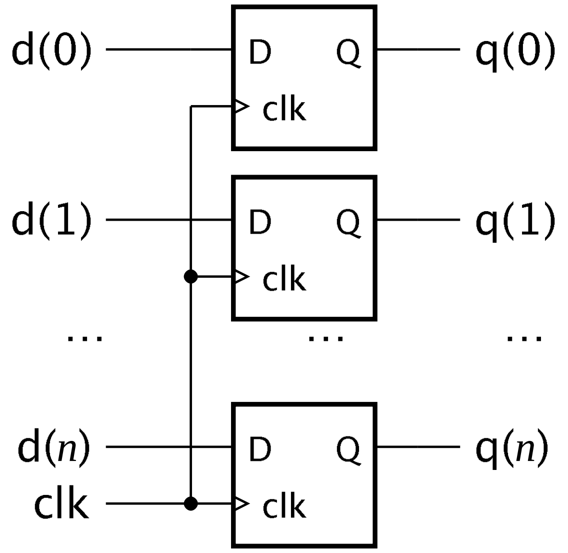
- Store a multi-bit encoded value
  - One D-flipflop per bit
  - Stores a new value on each clock cycle

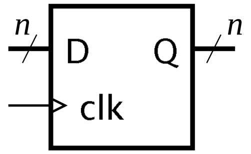
```
wire [n:0] d;
reg  [n:0] q;
...

always @(posedge clk)
  q <= d;
```

event list

nonblocking asignment

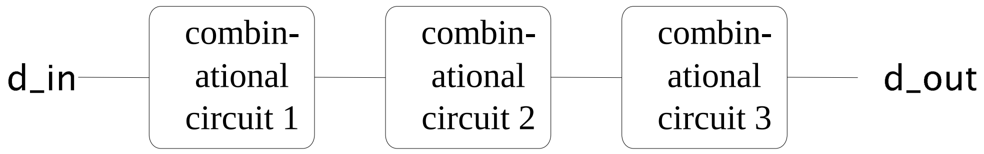


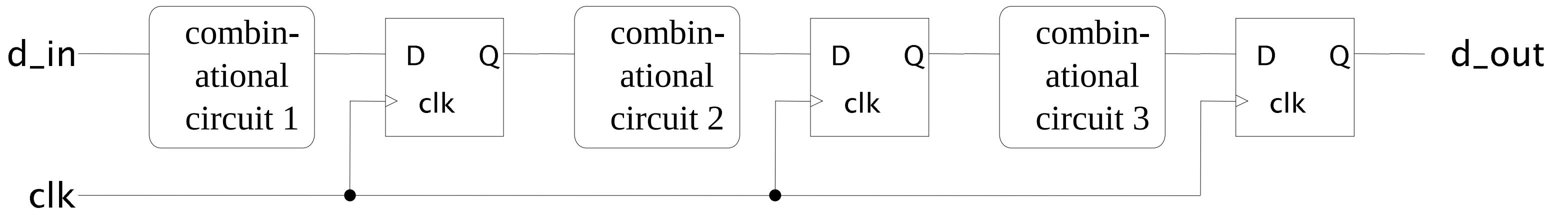


Pipelines Using Registers

Total delay = $\text{Delay}_1 + \text{Delay}_2 + \text{Delay}_3$

Interval between outputs > Total delay





Clock period = $\max(\text{Delay}_1, \text{Delay}_2, \text{Delay}_3)$

Total delay = $3 \times$ clock period

Interval between outputs = 1 clock period

Pipeline Example

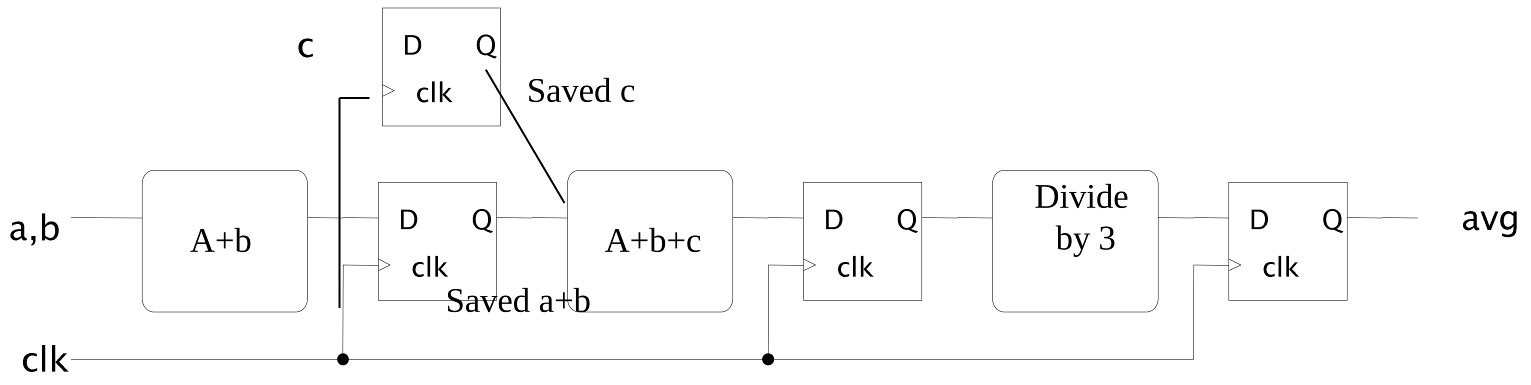
- Compute the average of corresponding numbers in three input streams
  - New values arrive on each clock edge

```
module average_pipeline ( output reg signed [0:13] avg,
                          input      signed [0:13] a, b, c,
wire signed [0:14] a_plus_b;
Wire signed [0:15] sum;
 wire signed [0:22] sum_div_3;
  reg  signed [0:14] saved_a_plus_b
Reg signed [0:13] saved_c
Reg [0:15] saved_sum;
  ...
input                    clk );
```

Pipeline Example

```
  ...
  assign a_plus_b = a + b;
  always @(posedge clk) begin  // Pipeline register 1
    saved_a_plus_b <= a_plus_b;
    saved_c        <= c;
  end
  assign sum = saved_a_plus_b + saved_c;
  always @(posedge clk)  // Pipeline register 2
    saved_sum <= sum;
  assign sum_div_3 = saved_sum * 7'b0101010;
  always @(posedge clk)  // Pipeline register 3
    avg <= sum_div_3;
endmodule
```

Blockdiagram


c
D
Q
clk
Saved c
a,b
A+b
D
Q
clk
Saved a+b
A+b+c
D
Q
clk
Divide by 3
D
Q
clk
avg
clk