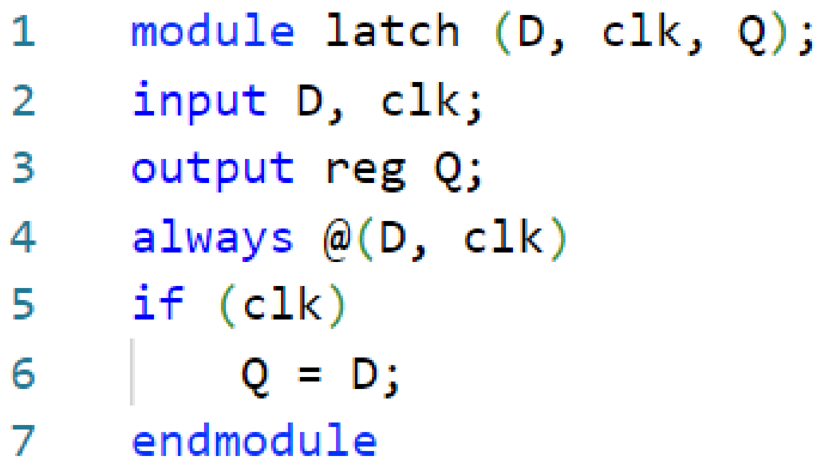
Verilog HDL:
Examples : Sequential Circuits

Pravin Zode

Outline

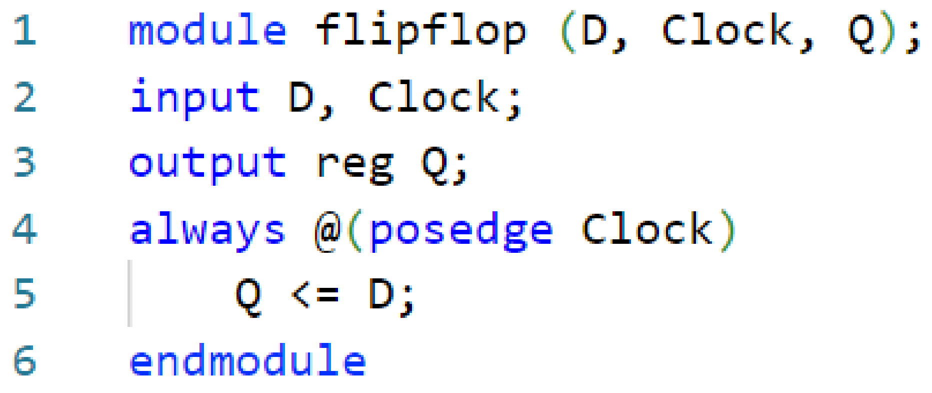
- D-Latch and D-FF

D-Latch & D-FF



```verilog
module latch (D, clk, Q);
input D, clk;
output reg Q;
always @(D, clk)
if (clk)
    Q = D;
endmodule
```

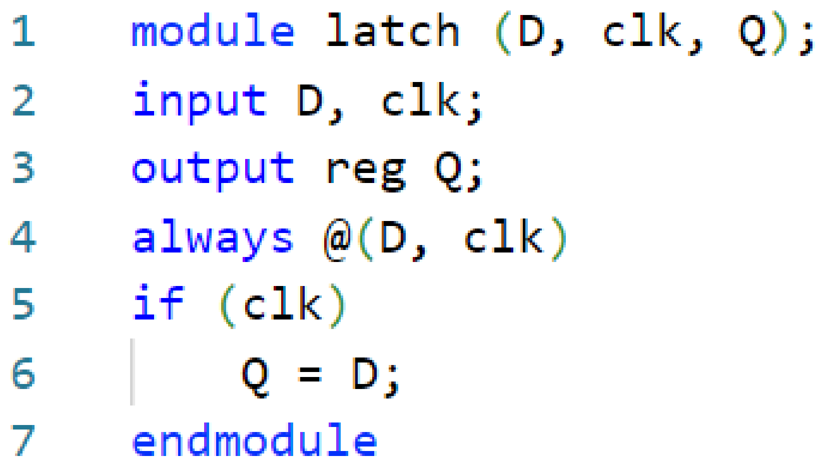
D-Latch



```verilog
module flipflop (D, Clock, Q);
input D, Clock;
output reg Q;
always @(posedge Clock)
    Q <= D;
endmodule
```

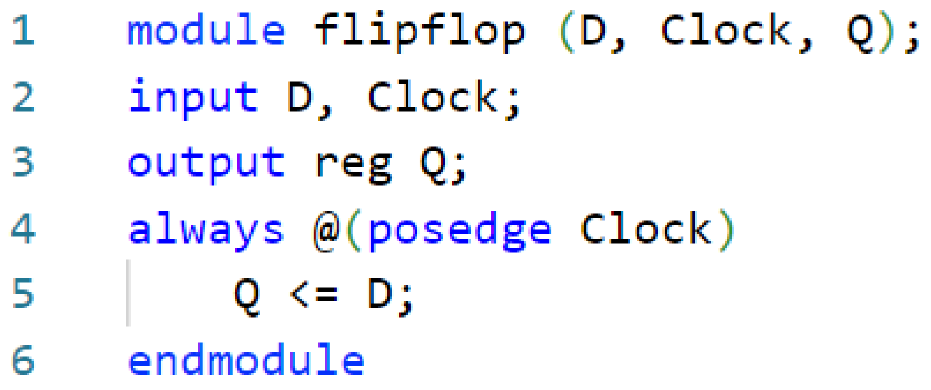
D-FlipFlop

D-Latch & D-FF



```verilog
module latch (D, clk, Q);
input D, clk;
output reg Q;
always @(D, clk)
if (clk)
    Q = D;
endmodule
```

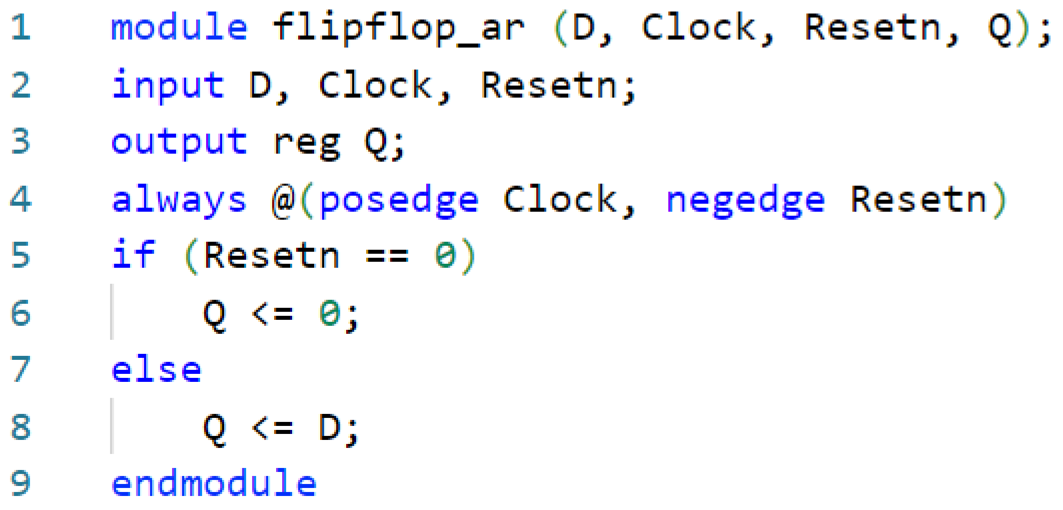
D-Latch



```verilog
module flipflop (D, Clock, Q);
input D, Clock;
output reg Q;
always @(posedge Clock)
    Q <= D;
endmodule
```

D-FlipFlop

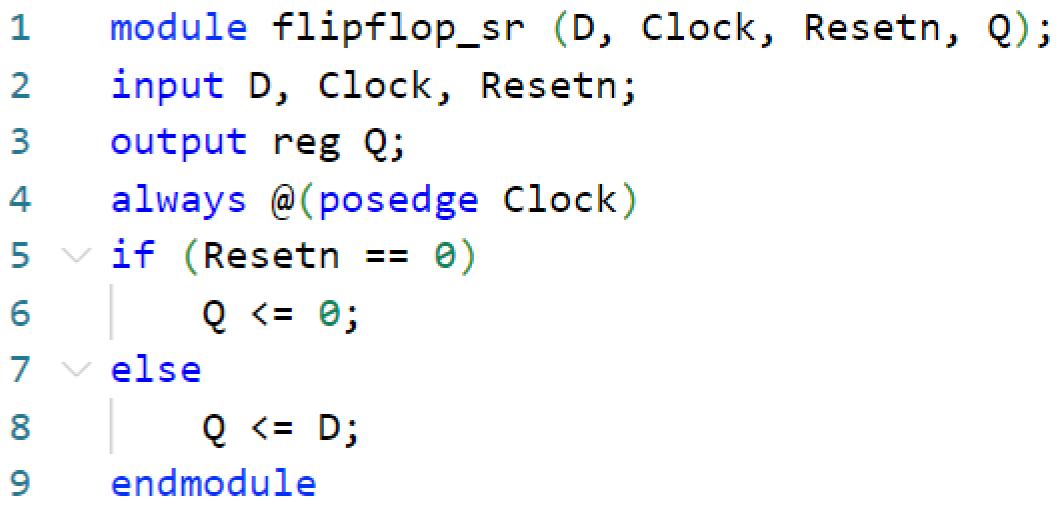
Flip-Flop Sychronous & Asynchronous Reset



```
module flipflop_ar (D, Clock, Resetn, Q);
input D, Clock, Resetn;
output reg Q;
always @(posedge Clock, negedge Resetn)
if (Resetn == 0)
    Q <= 0;
else
    Q <= D;
endmodule
```

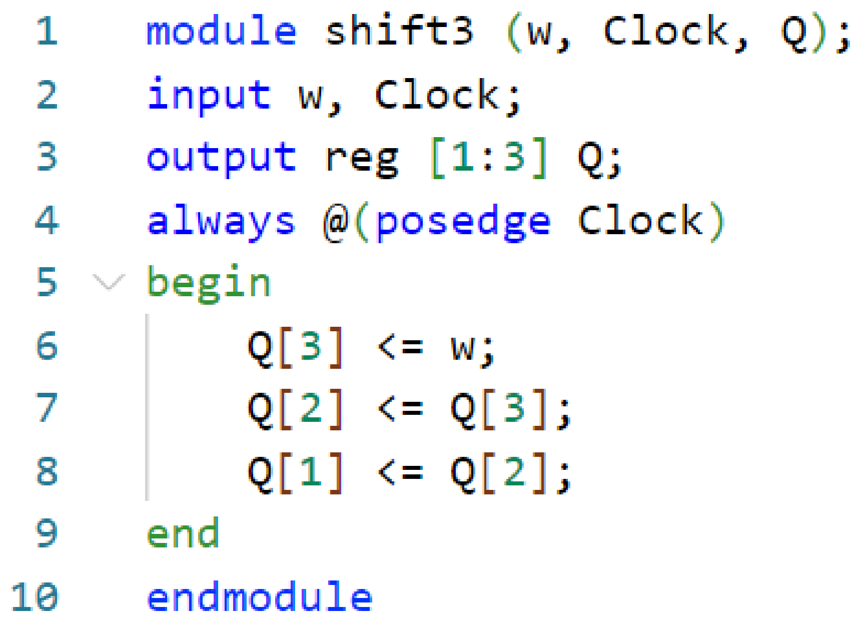
D flip-flop with synchronous reset

D flip-flop with asynchronous reset

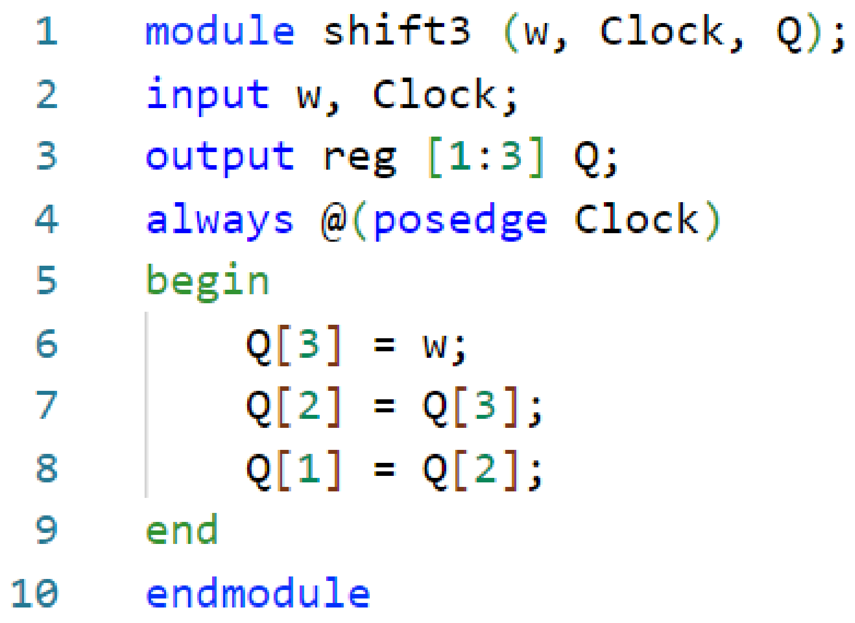


```
module flipflop_sr (D, Clock, Resetn, Q);
input D, Clock, Resetn;
output reg Q;
always @(posedge Clock)
if (Resetn == 0)
    Q <= 0;
else
    Q <= D;
endmodule
```

Shift Register



```verilog
module shift3 (w, Clock, Q);
input w, Clock;
output reg [1:3] Q;
always @(posedge Clock)
begin
    Q[3] <= w;
    Q[2] <= Q[3];
    Q[1] <= Q[2];
end
endmodule
```



```verilog
module shift3 (w, Clock, Q);
input w, Clock;
output reg [1:3] Q;
always @(posedge Clock)
begin
    Q[3] = w;
    Q[2] = Q[3];
    Q[1] = Q[2];
end
endmodule
```

Three-bit shift register

Wrong code for a three-bit shift register

Shift Register

```verilog
module shiftreg_4bit (clock, clear, A, E);
input clock, clear, A;
output reg E;
reg B, C, D;
always @(posedge clock or negedge clear)
begin
if (!clear)
begin
    B<=0; C<=0; D<=0; E<=0;
end
else
begin
    E <= D;
    D <= C;
    C <= B;
    B <= A;
end
end
endmodule
```

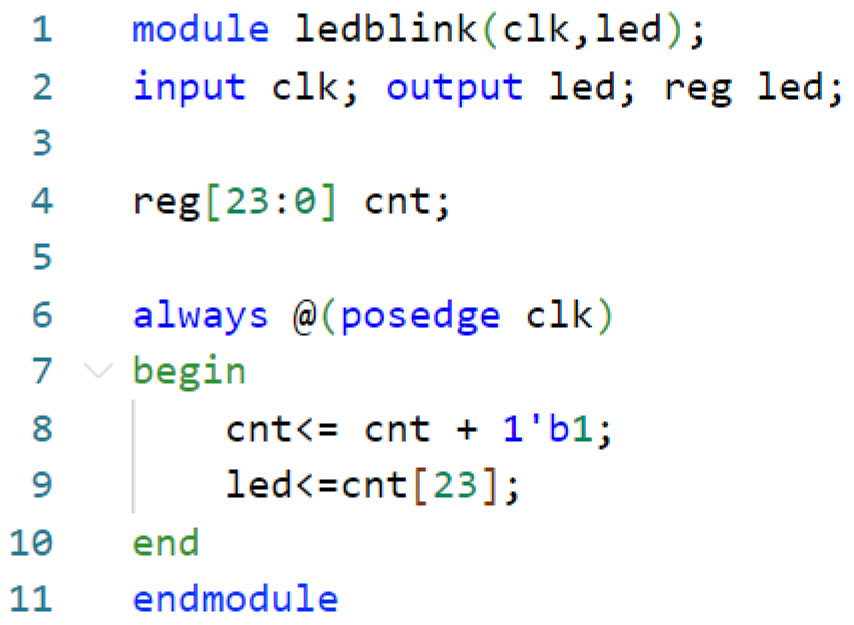
```verilog
module shiftreg_4bit_tb;
reg clk, clr, in; wire out; integer i;
shiftreg_4bit SR (clk, clr, in, out);
initial
begin clk = 1'b0; #2 clr = 0; #5 clr = 1;
end
always #5 clk = ~clk;
initial begin #2;
repeat (2)
begin #10 in=0; #10 in=0; #10 in=1; #10 in=1;
end
end
initial
begin
$dumpfile ("shifter.vcd");
$dumpvars (0, shift_test);
#200 $finish;
end
endmodule
```

Counter

```verilog
module counter (clear, clock, count);
parameter N = 7;
input clear, clock;
output reg [0:N] count;
always @(negedge clock)
if (clear)
    count <= 0;
else
    count <= count + 1;
endmodule
```

```verilog
module test_counter;
reg clk, clr;
wire [7:0] out;
counter CNT (clr, clk, out);
initial clk = 1'b0;
always #5 clk = ~clk;
initial
begin
clr = 1'b1;
#15 clr = 1'b0;
#200 clr = 1'b1;
#10 $finish;
end
initial
begin
$dumpfile ("counter.vcd");
$dumpvars (0, test_counter);
$monitor ($time, " Count: %d", out);
end
endmodule
```

Clock Divider



```verilog
module ledblink(clk,led);
input clk; output led; reg led;

reg[23:0] cnt;

always @(posedge clk)
begin
    cnt<= cnt + 1'b1;
    led<=cnt[23];
end
endmodule
```

https://www.fpga4fun.com/Opto.html

https://eecs.blog/max-ii-cpld-basic-getting-started-tutorial/

Thank you !

Happy Learning