# Digital System Design with HDL (I)

## Lecture 13

Dr. Ming Xu

Dept of Electrical & Electronic Engineering

XJTLU



## In This Session

- Design Example — Bus Structure

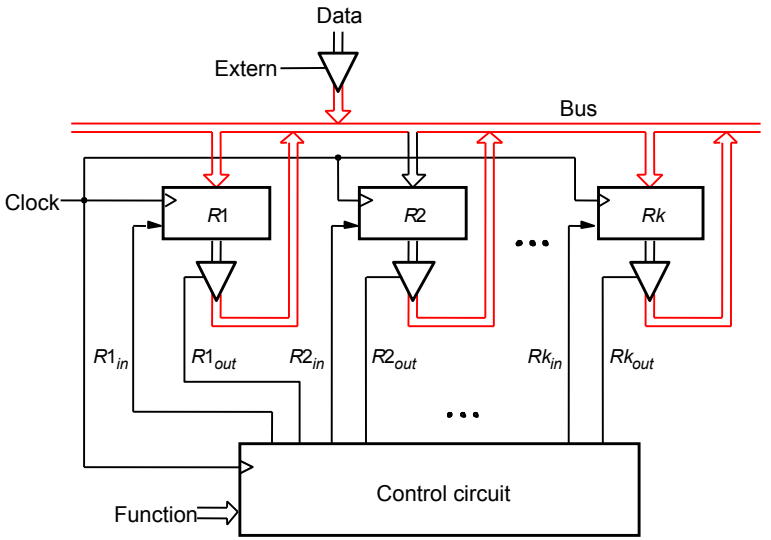


## The Functions

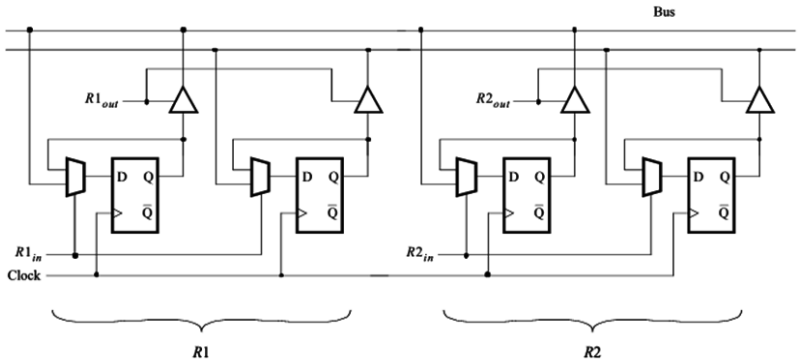
- A digital system contains a set of registers used to store data.
- Each register is connected to a bus, which is used to transfer data into and out of the register.
- Data can be also placed on the bus from an external circuit block.
- At any time only one register or the external block can put data onto the bus.



## Bus Structure for Digital Systems



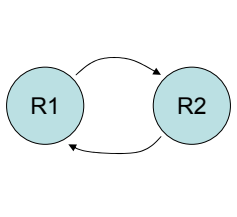
# Connecting Registers to a Bus

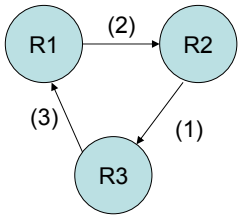



2-bit bus connecting: 2 registers

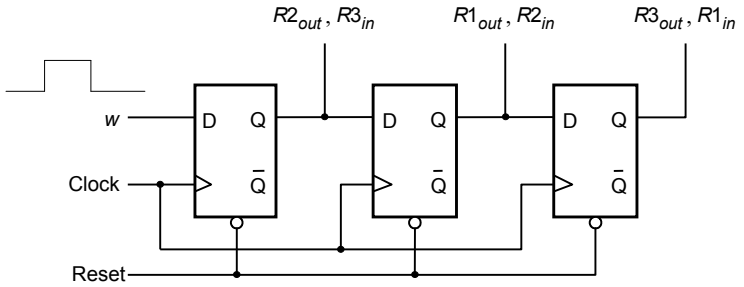


# Swapping operation



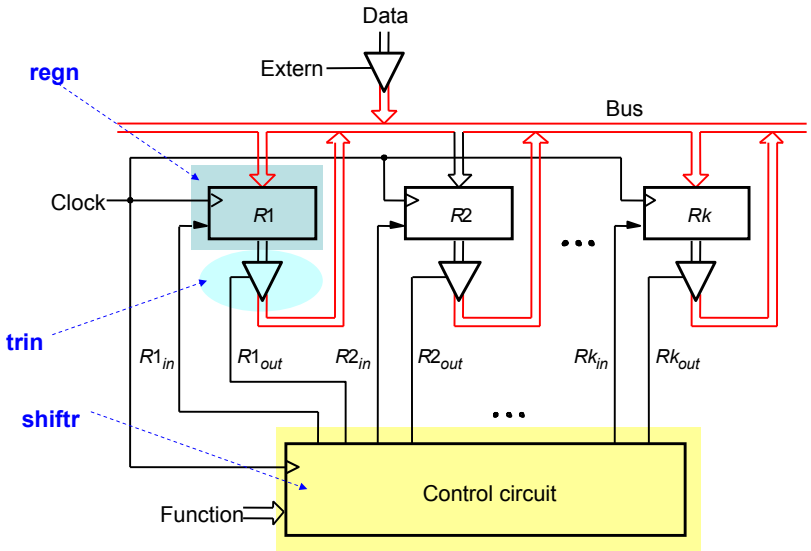




Using register 3 for temporary storage





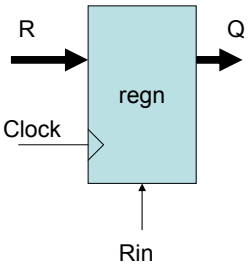

# Verilog for Swapping





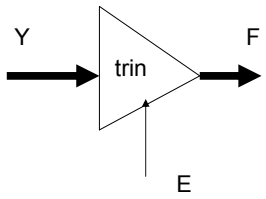

# N-bit Register Module

```
module regn (R, Rin, Clock, Q);
    parameter n = 8;
    input [n-1:0] R;
    input Rin, Clock;
    output [n-1:0] Q;
    reg [n-1:0] Q;

    always @(posedge Clock)
        if (Rin)
            Q <= R;

endmodule
```



# Tri-State Module

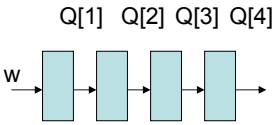
```
module trin (Y, E, F);
    parameter n = 8;
    input [n-1:0] Y;
    input E;
    output [n-1:0] F;
    wire [n-1:0] F;

    assign F = E ? Y : 'bz;

endmodule
```





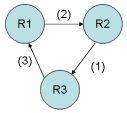

# Control Circuit Module

```
module shiftr (Resetn, w, Clock, Q);
    parameter m = 4;
    input Resetn, w, Clock;
    output [1:m] Q;
    reg [1:m] Q;
    integer k;

    always @(negedge Resetn or posedge Clock)
        if (!Resetn)
            Q <= 0;
        else
        begin
            for (k = m; k > 1; k = k-1)
                Q[k] <= Q[k-1];
            Q[1] <= w;
        end
endmodule
```






# A Digital System for Swap




```
module swap (Data, Resetn, w, Clock,
            Extern, RinExt, BusWires);

    input [7:0] Data;
    input Resetn, w, Clock, Extern;
    input [1:3] RinExt;
    output [7:0] BusWires;
    tri [7:0] BusWires;
    wire [1:3] Rin, Rout, Q;
    wire [7:0] R1, R2, R3;

    shiftr control (Resetn, w, Clock, Q);
            defparam control.m = 3;
```

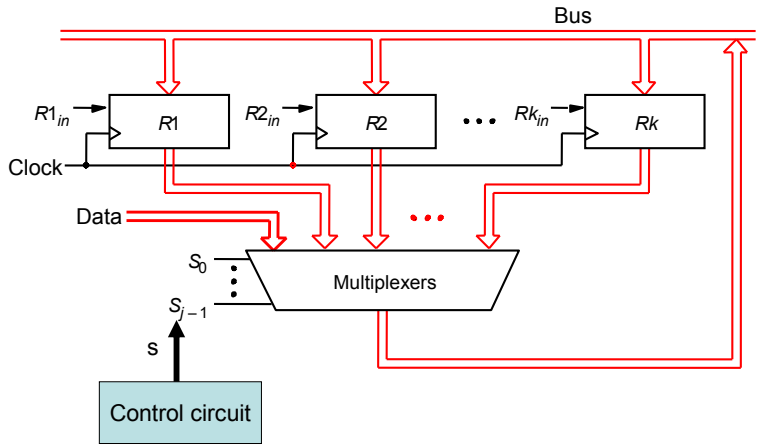
```
    assign Rin[1] = RinExt[1] | Q[3];
    assign Rin[2] = RinExt[2] | Q[2];
    assign Rin[3] = RinExt[3] | Q[1];
    assign Rout[1] = Q[2];
    assign Rout[2] = Q[1];
    assign Rout[3] = Q[3];

    regn reg_1 (BusWires, Rin[1], Clock, R1);
    regn reg_2 (BusWires, Rin[2], Clock, R2);
    regn reg_3 (BusWires, Rin[3], Clock, R3);

    trin tri_ext (Data, Extern, BusWires);
    trin tri_1 (R1, Rout[1], BusWires);
    trin tri_2 (R2, Rout[2], BusWires);
    trin tri_3 (R3, Rout[3], BusWires);

endmodule
```

Q[1]: $R2_{out}$, $R3_{in}$
Q[2]: $R1_{out}$, $R2_{in}$
Q[3]: $R3_{out}$, $R1_{in}$

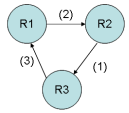
RinExt [i] allows external data to be loaded into Ri



# Swap Based on Multiplexers



# Swap using multiplexers




```
module swapmux (Data, Resetn, w, Clock,
RinExt, BusWires);
    input [7:0] Data;
    input Resetn, w, Clock;
    input [1:3] RinExt;
    output [7:0] BusWires;
    reg [7:0] BusWires;
    wire [1:3] Rin, Q;
    wire [7:0] R1, R2, R3;

    shiftr control (Resetn, w, Clock, Q);
        defparam control.m = 3;

    assign Rin[1] = RinExt[1] | Q[3];
    assign Rin[2] = RinExt[2] | Q[2];
    assign Rin[3] = RinExt[3] | Q[1];
```

```
regn reg_1 (BusWires, Rin[1], Clock, R1);
regn reg_2 (BusWires, Rin[2], Clock, R2);
regn reg_3 (BusWires, Rin[3], Clock, R3);

always @(Q or Data or R1 or R2 or R3)
  begin
    if (Q == 3'b000)     BusWires = Data;
    else if (Q == 3'b100) BusWires = R2;
    else if (Q == 3'b010) BusWires = R1;
    else      BusWires = R3;
  end

endmodule
```

Q[1]: $R2_{out}$, $R3_{in}$

Q[2]: $R1_{out}$, $R2_{in}$

Q[3]: $R3_{out}$, $R1_{in}$

Q [1:3]: Q[1], Q[2], Q[3]

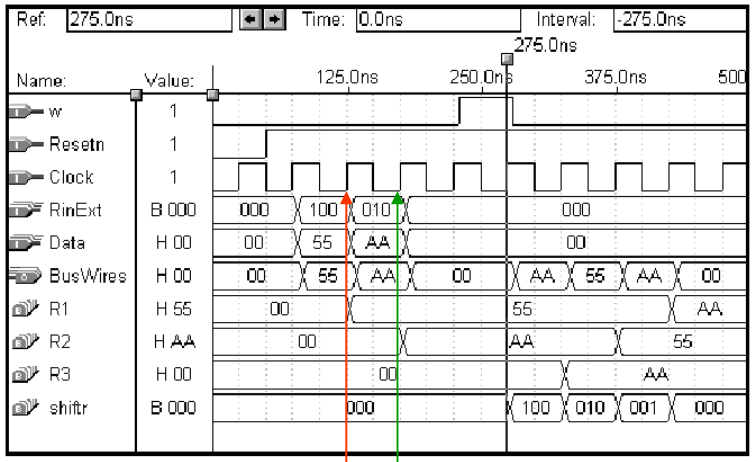
Q[1]: $R2_{out}$

Q[2]: $R1_{out}$

Q[3]: $R3_{out}$



# Simulation



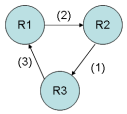

w, RinExt, Data are generated elsewhere but aligned with Clock.

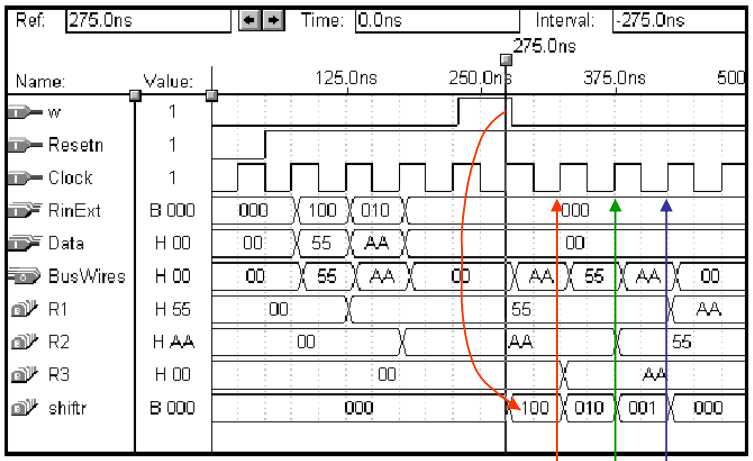
AA written into R2

55 written into R1



# Simulation




R3 => R1

R1 => R2

R2 => R3