NMK20603

Computer Architecture

MY1

Codes so far:

- ✔ alu_4b

MY1

Homework?

⇒ zbuff (>1b)

⇒ decode38

⇒ cmp_2b

Digital systems:

- combinational

  $\Rightarrow$ logic gates (and/or/inv)

- sequential

  = latch/flip-flop

MY1

Let's go

sequential

MY1

Describing Sequential Logic

MY1

Latches & Flip-flop

MY1

Difference?

MY1

Trigger:

Level vs Edge!

Keyword

- always

⇒ like initial

MY1

```
always
begin
// code goes here
// data type: reg
// 'always' triggered
end
```

Syntax

- @()

⇒ sensitivity list

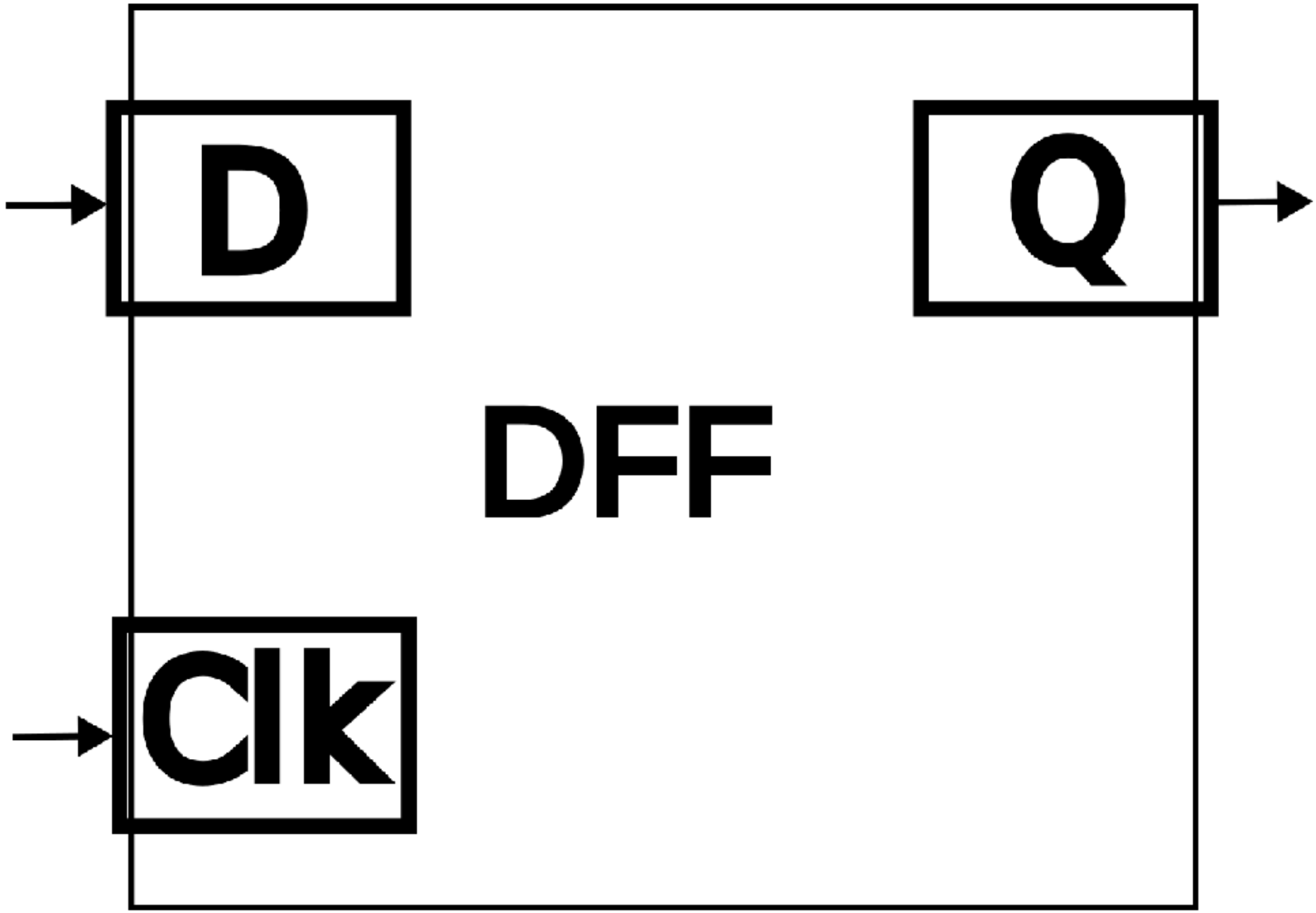
⇒ v95: or separated

⇒ v2k1: comma separated

```
always @(A or B)
begin
// triggered when
// A/B changes
end
```

D Flip-flop

MY1



Describe:

dff

MY1

```
module dff (iclk,idat,odat);
input iclk,idat;
output odat;
reg odat;
always @(iclk) begin
  if (iclk==1'b1) begin
    odat = idat;
  end
end
endmodule
```

MY1

Syntax

- reg assignment

⇒ = vs <=

⇒ sequence vs concurrent

```
always @(idat) begin
   tdat = idat
   odat = tdat;
   // odat always same as tdat
end
```

MY1

```
always @(idat) begin
   tdat <= idat
   odat <= tdat;
   // odat may differ from tdat
end
```

MY1

Testbench?

⇒ no 'template'

Clock 'generator'

MY1

```
// use in tb
// 10TU clock period
parameter CLKP = 10;
reg dclk; // clock net
always begin
  #(CLKP/2) dclk = ~dclk;
end
```

MY1

User task

@custom sequence!

Keyword

- task/endtask

⇒ for tb only

```
// assume @+ve edge dff
// assume driver net is ddat
// should call this @-ve edge
reg ddat;
task dff_wr ;
  input integer dbit;
  begin
    ddat = dbit;
    $display("-- [%g] WR-dff:'%b' ",$time,ddat);
    #(CLKP/2); // setup time
    // should latch here
    #(CLKP/2); // hold time
    // 1-cycle
  end
endtask
```

DFF output 'monitor'

MY1

```
// detect dff output
// can also use mdat!
always @(dut.odat) begin
  $write("[%05g] dffQ=%b\n",$time,dut.odat);
end
```

Notice:

- dut.odat?

MY1

Describe:

dff_tb

MY1

```
module dff_tb ();
// clock generator
// task to write to dff
// dff output monitor
// the rest...
wire mdat;
initial begin
  // reset condition
  dclk = 1'b1; //at tu=0, dclk is 1
  ddat = 1'b0;
  // start test
  #(CLKP/2); // wait for -ve edge
  dff_wr(1);
  #CLKP;
  // verify here?
  #CLKP;
  dff_wr(0);
  #(CLKP*5);
  $stop;
end
dff dut (dclk,ddat,mdat);
endmodule
```

Register?

- dff with purpose!

⇒ wr/rd on request

Ports:

- ✔ iclk, idat, odat
- ✔ wenb, renb (write/read enable)

MY1

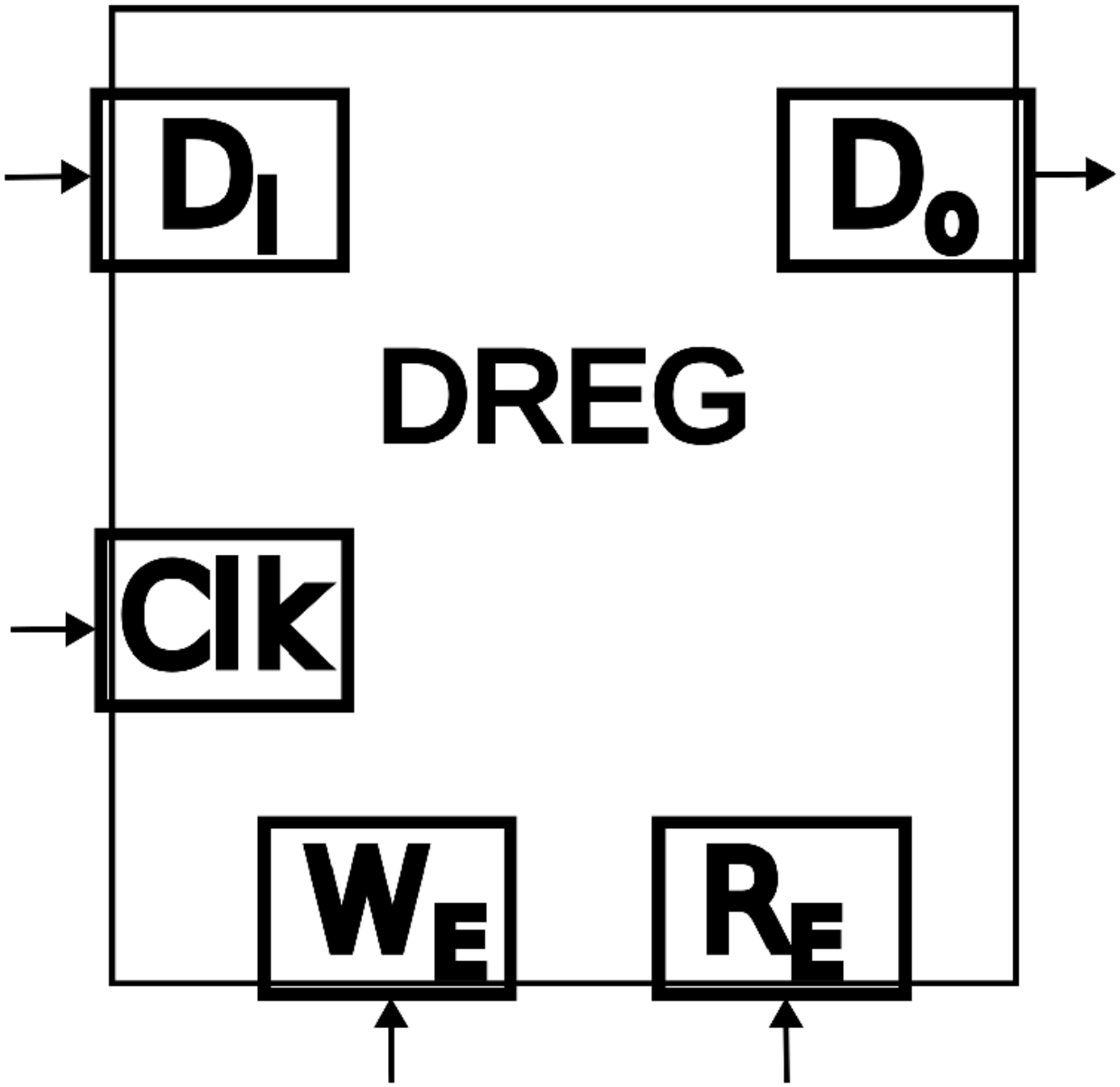
Syntax

- posedge

⇒ sensitivity list

```
always @(posedge dclk)
begin
// triggered when
// dclk transition 0->1
end
```

MY1



MY1

Describe:

dreg

⇒ BITS=4

```
module dreg (iclk,wenb,renb,idat,odat);
parameter BITS=4;
input iclk,wenb,renb;
input[BITS-1:0] idat;
output[BITS-1:0] odat;
wire[BITS-1:0] odat
output[BITS-1:0] tdat;
always @(posedge iclk) begin
  if (wenb==1'b1) begin
    odat = idat;
  end
end
defparam oenb.BITS = BITS;
zbuff oenb (renb,tdat,odat);
endmodule
```

Practice:

⇒ task>dreg_write

```
// modified from dff_wr
// assume wr_enb drive signal dwen
// using parameter BITS
reg dwen;
reg[BITS-1:0] ddat;
task dreg_write ;
  input integer dval;
  begin
    ddat = dval; // setup data for write
    dwen = 1'b1; // enable write for 1 clk period
    $display("-- [%05g] WR-dreg:'%b' ",$time,ddat);
    #(CLKP/2); // setup time
    #(CLKP/2); // hold time
    dwen = 1'b0; // disable write
  end
endtask
```

Practice:

⇒ 'monitor' code for dreg?

```
// similar to that in dff_tb
// monitor reg changes (NOT output)
always @(dut.tdat) begin
  $write("## [%05g] dregQ=%b\n",$time,dut.tdat);
end
```

Practice:

⇒ task>dreg_read

```
// assume rd_enb drive signal dren
// assume output net is mdat
// assume read on +ve
// call on -ve clk edge, mutual exclusive to dreg_wr
reg dren;
wire[BITS-1:0] mdat;
task dreg_read ;
  begin
    dren = 1'b1; // enable read for 1 clk period
    #(CLKP/2); // setup time
    $display("@@ [%05g] RD-dreg:'%b' ",$time,mdat);
    #(CLKP/2); // hold time
    dren = 1'b0; // disable read
  end
endtask
```

Practice:

⇒ reset condition?

```
dwen = 1'b0;
dren = 1'b0;
```

MY1

Practice:

dreg_tb

MY1

```
module dreg_tb ();
parameter BITS=4;
// clock generator
// task:dreg_write
// dreg output monitor
// task:dreg_read
// the rest...
initial begin
  dclk = 1'b1; //ddat = 4'h5;
  dwen = 1'b0; dren = 1'b0;
  // start test
  dreg_read();
  #(CLKP/2); // wait for -ve edge
  dreg_write(4'b1010);
  #CLKP;
  dreg_read();
  dreg_write(12);
  #CLKP;
  dreg_read();
  $stop;
end
defparam dut.BITS = BITS;
dreg dut (dclk,dwen,dren,ddat,mdat);
endmodule
```

Register block?

- 4x4b register
- with bus to move!

⇒ what else do we need?

MY1

Homework:

⇒ regblock4

MY1