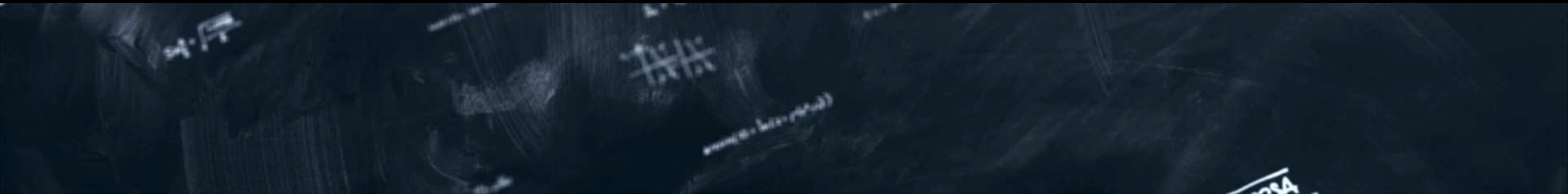
Register Transfer Level Programming

ECE 2372 Modern Digital System Design | Texas Tech University



Lecture Overview

- What is Register Transfer Level?
- RTL Syntax in Verilog HDL
- RTL Examples

What is Register Transfer Level?

Register Transfer Level Programming | Modern Digital System Design

What is Register Transfer Level?

- Register Transfer Level (RTL) is a design abstraction which models the flow of digital signals between registers and the logical operations performed on those signals.
- Most digital systems are designed at the RTL level.

```
 8  module ff_d(D, CLK, Q);
 9
10      // DATA INPUTS
11      input D;
12
13      // CONTROL INPUTS
14      input CLK;
15
16      // OUTPUT REGISTERS
17      output reg Q;
18
19      // RTL CIRCUIT IMPLEMENTATION
20      always @(posedge CLK)
21      begin
22          Q <= D;
23      end
24
25  endmodule
```

What is Register Transfer Level?

- RTL programming is very similar to computer programming.
- Similar structures are used
  - If/else
  - Loops

- Verilog is still not a programming language.
- It is common for new engineers to try and use RTL like C/C++.

What questions do you have?

EXAM QUESTION: Register Transfer Level

Which of the following RTL commands defines a 6-bit output register?

A. `reg[5:0] myReg;`

B. `output [5:0] myReg;`

C. `output reg [5:0] myReg;`

D. `output reg [6:0] myReg;`

EXAM QUESTION: Register Transfer Level

Which of the following RTL commands defines a 6-bit output register?

~~A. reg[5:0] myReg;~~

~~B. output [5:0] myReg;~~

C. output reg [5:0] myReg;

~~D. output reg [6:0] myReg;~~

RTL Syntax in Verilog HDL

Register Transfer Level Programming | Modern Digital System Design

RTL Syntax in Verilog HDL

- Registers
- Trigger
- If/Else Syntax in Verilog
- For Loops in Verilog
- While Loops in Verilog
- Non-blocking Assignment

RTL Syntax: Registers

- Registers are provided in Verilog HDL.
- Usually (but not always), you'll declare your output ports as registers.

```
16 |     // OUTPUT REGISTERS
17 |     output reg Q;
```

RTL Syntax: Registers

- Registers can be declared to be N-bit long.

```
30   output reg [7:0] Accumulator;
```

RTL Syntax: Trigger

Register transfers are triggered using the `always@` command.

```
// RTL CIRCUIT IMPLEMENTATION
always @(posedge CLK)
begin
    Q <= D;
end
```

RTL Syntax: Trigger

```
27   always@(posedge CLK) // Rising Edge of CLK
28
29   always@(negedge CLK) // Falling Edge of CLK
30
31   always@(CLK) // Rising and Falling Edge of CLK
32
33   always@(*) // Any Rising or Falling Edge of ANY Input.
```

RTL Syntax: IF/ELSE

```
27    if (CONDITION)
28    begin
29        // EXECUTE IF TRUE
30    end
31    else
32    begin
33        // EXECUTE IF FALSE
34    end
```

RTL Syntax: FOR Loop

```
27    for (START; END_CONDITION; CHANGE)
28    begin
29        // EXECUTE
30    end
```

RTL Syntax: WHILE Loop

- While loops aren't commonly used, but they are available.

```
27    while (CONDITION)
28    begin
29        // EXECUTE WHILE CONDITION
30        // IS TRUE
31    end
```

Blocking vs Nonblocking Assignment

Verilog supports two separate types of assignment.

- **Blocking –** Will hold all execution until complete (synchronous)
- **Nonblocking –** Will not hold up execution (asynchronous)

```
27    // Nonblocking assignment
28    Q = D;
29
30    // Blocking Assignment
31    Q <= D;
```

RTL Syntax in Verilog HDL

- Registers
- Trigger
- If/Else Syntax in Verilog
- For Loops in Verilog
- While Loops in Verilog
- Non-blocking Assignment

What questions do you have?

EXAM QUESTION: RTL Syntax

Which of the following RTL commands will cause the logic instructions to trigger on the falling edge of `CLK`?

A. `always@(posedge CLK);`

B. `always@(negedge CLK);`

C. `always@(CLK);`

D. `always@(*);`

EXAM QUESTION: RTL Syntax

Which of the following RTL commands will cause the logic instructions to trigger on the falling edge of `CLK`?

~~A. always@(posedge CLK);~~

B. always@(negedge CLK);

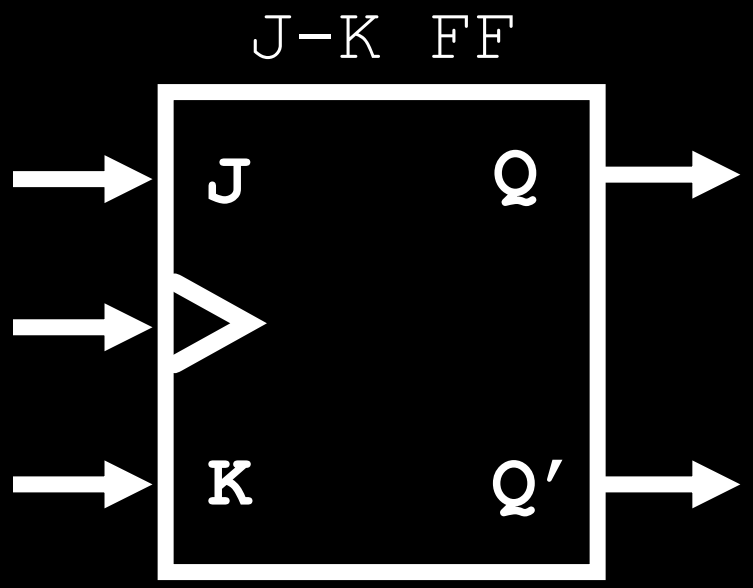
~~C. always@(CLK);~~

~~D. always@(*);~~

RTL Examples

Register Transfer Level Programming | Modern Digital System Design

RTL Example: J-K Flip-Flop



Input	Output
J = K = 0	No Change
J = 1, K = 0	SET (After CLK Edge)
J = 0, K = 1	RESET (After CLK Edge)
J = K = 1	TOGGLE (After CLK Edge)

```
 8  module ff_jk(J, K, CLK, Q);
 9      // DATA INPUTS
10      input J, K;
11
12      // CONTROL INPUTS
13      input CLK;
14      // OUTPUT REGISTERS
15      output reg Q = 1'b0;
16
17      // RTL LOGIC IMPLEMENTATION
18      always @(posedge CLK)
19      begin
20          Q <= J & ~Q | ~K & Q;
21      end
22
23  endmodule
```

Test Benches in RTL

There are only two main things that are different about running a test bench on an RTL module.

1. Setup the clock
2. Manually stop the test.

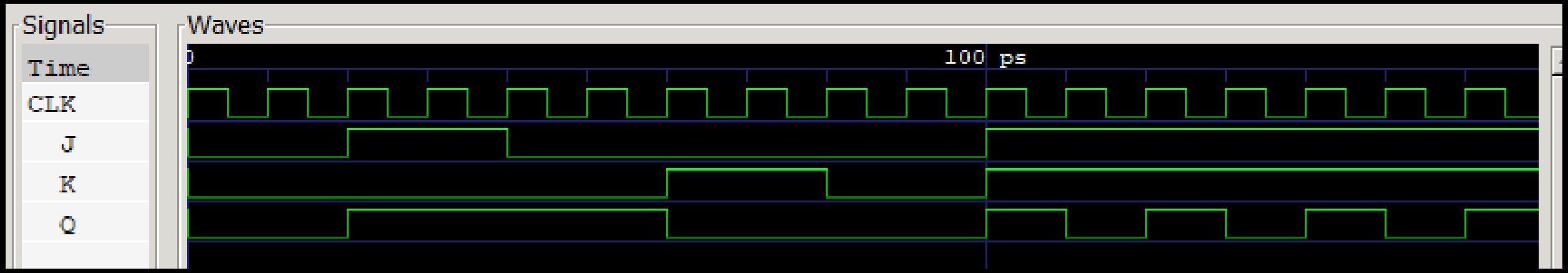
Setup the Clock

```
// CLOCK
reg CLK = 1'b0;
always
begin
    CLK = ~CLK;
    #5;
end
```

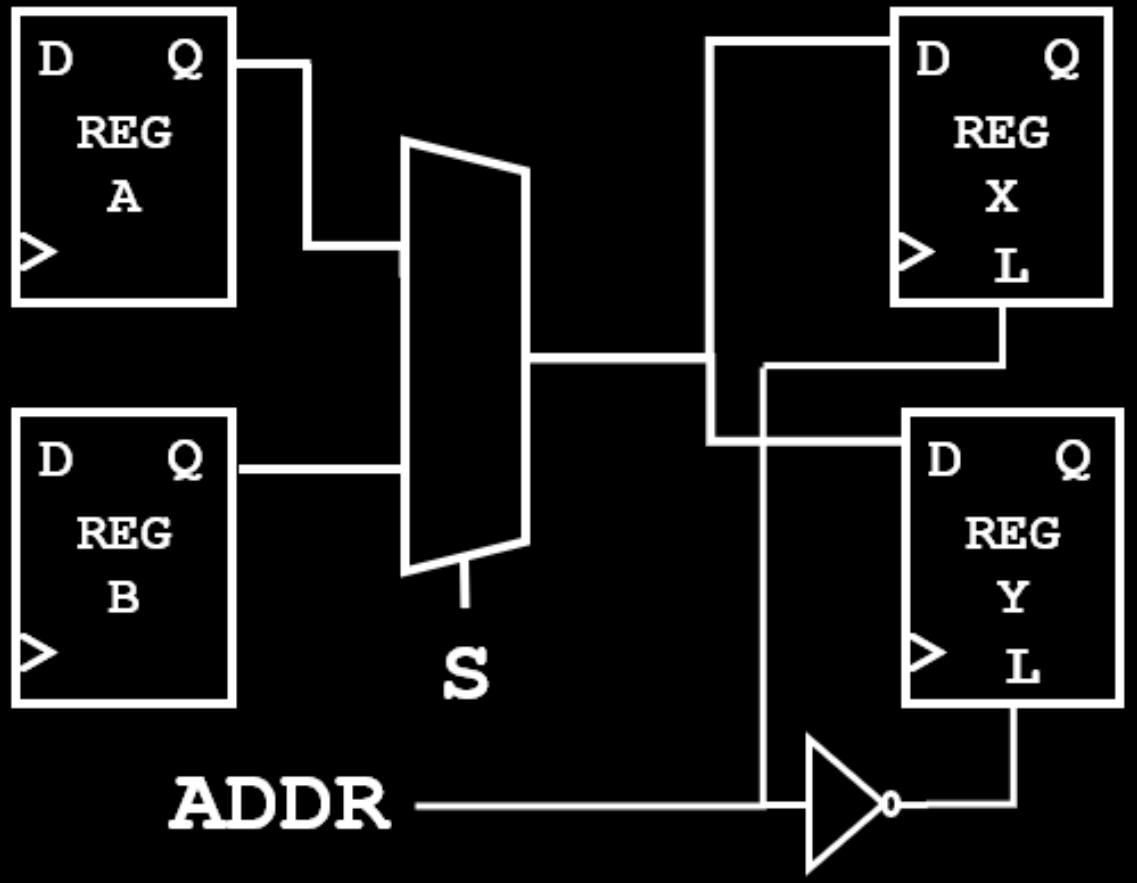
Manually stop the test.

- If you don’t stop the test manually, the clock will keep the test running forever.
- Stop the test with the `$finish` command.

```
// TEST EXECUTION
initial begin
    $dumpfile("ff_jk.vcd");
    $dumpvars(0, ff_jk_test);
    {J, K} = 2'b00; #20;
    {J, K} = 2'b10; #20;
    {J, K} = 2'b00; #20;
    {J, K} = 2'b01; #20;
    {J, K} = 2'b00; #20;
    {J, K} = 2'b11; #20;
    #50;
    $finish;
end
```


Signals
Waves
Time
CLK
J
K
Q
100 ps

RTL Example: Register Transfer



```
 9  module reg_tran(D, S, ADDR, CLK, Q);
10      // DATA INPUTS
11      input [1:0] D;
12      // CONTROL INPUTS
13      input S, ADDR, CLK;
14      // DATA OUTPUTS
15      output [1:0] Q;
16      // INTERNAL REGISTERS
17      reg rA, rB, rX, rY;
```

```
19      // RTL LOGIC IMPLEMENTATION
20      always@(posedge CLK) begin
21          rA <= D[1];
22          rB <= D[0];
23
24          if(S == 0) begin
25              if (ADDR == 1) begin
26                  rX <= rA;
27              end else begin
28                  rY <= rA;
29              end
30          end else begin
31              if (ADDR == 1) begin
32                  rX <= rB;
33              end else begin
34                  rY <= rB;
35              end
36          end
37      end
```

```
39      // COMBINATIONAL LOGIC IMPLEMENTATION
40      assign Q[1] = rX;
41      assign Q[0] = rY;
```

RTL Examples: ALU Project

RTL Syntax drastically simplifies the implementation from the ALU project.

```
 9    // DATA INPUTS
10    input [7:0] A, B;
11
12    // CONTROL INPUTS
13    input S, E;
14
15    // OUTPUT REGISTERS
16    output reg [7:0] W;
17
```

```
18      // RTL LOGIC IMPLEMENTATION
19      always@(*) begin
20          if (E == 1) begin
21              if (S == 0) begin
22                  W <= A + B;
23              end else begin
24                  W <= A - B;
25              end
26          end else begin
27              W <= 0;
28          end
29      end
```

What questions do you have?

Lecture Recap

- What is Register Transfer Level?
- RTL Syntax in Verilog HDL
- RTL Examples

What questions do you have?

Register Transfer Level Programming

ECE 2372 Modern Digital System Design | Texas Tech University