Embedded Systems

CS429 Lab8

Name: Dipean Dasgupta ID:202151188

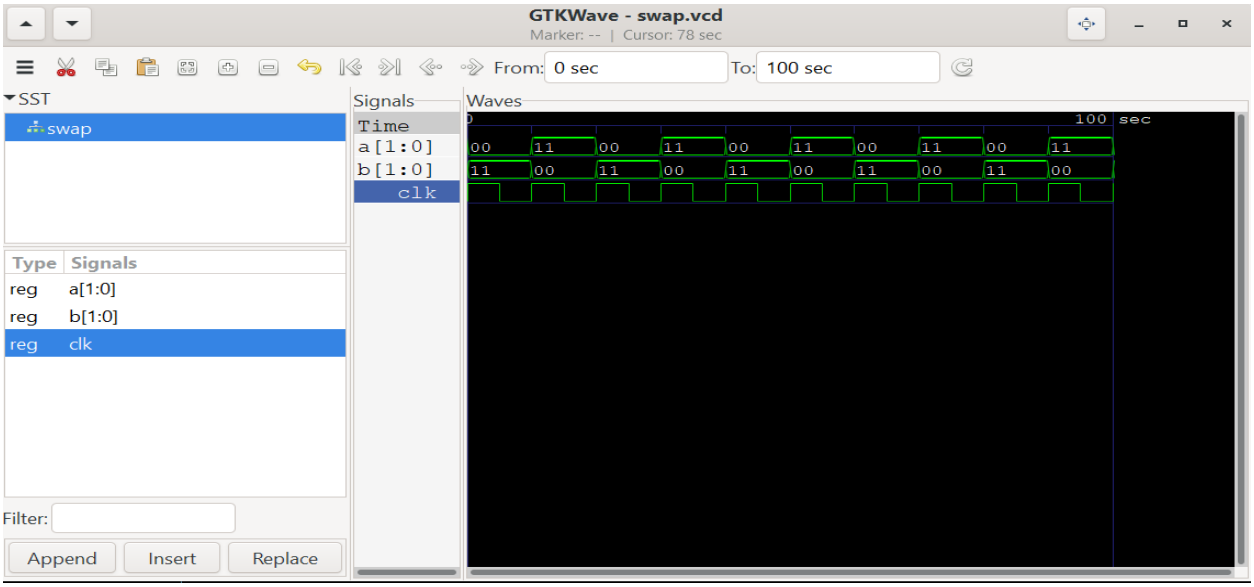
Task1: An embedded system has two output terminals as A and B. The signal on A and B are of 2 bits each and are to be swapped continuously. Implement the system design in Verilog HDL to execute the task.

Design code:

```
module swap(
    output reg clk,
    output reg [1:0]a,b
);
initial begin
    clk=1'b1;
    a=2'b00;
    b=2'b11;
    $dumpfile("swap.vcd");
    $dumpvars(0,swap);
#100 $finish;
end
always #5 clk = ~clk;
always @(posedge clk )
begin
    b <= a;
    a <= b;
    $display("A=%b B=%b", a, b);
end
endmodule
```

OUTPUT:

```
[Running] lab9_1.v
VCD info: dumpfile swap.vcd opened for output.
A=00 B=11
A=11 B=00
A=00 B=11
A=11 B=00
A=00 B=11
A=11 B=00
A=00 B=11
A=11 B=00
A=00 B=11
lab9_1.v:11: $finish called at 100 (1s)
```



Task2: D latch and Flip-flops are often used to store the output of a system. Execute D latch and flip-flop in Verilog.

Latch:

```
`timescale 1ns/1ns
module latch(
    input d,en,
    output q
);
assign q=en?d:q;

endmodule

module latch_tb;
  parameter CLK_PERIOD = 10;
  reg d;
  reg en;
  reg clk;
  wire q;

  latch uut (
    .d(d),
    .en(en),
    .q(q)
  );
  always #((CLK_PERIOD)/2) clk = ~clk;
  // Stimulus
  initial begin
    d = 0; en = 0; clk = 0;
```

```
    #10 d = 1;
    #10 en = 1;
    #10 d = 0;
    #10 en = 0;
    #10 d = 1;
    #10 en = 1;
    #10;
    $finish;
  end
  always @(posedge clk) begin
    $display("q = %b", q);
  end
endmodule
```

OUTPUT:

```
[Running] lab9_2A.v
q = x
q = x
q = 1
q = 0
q = 0
q = 0
q = 1
lab9_2A.v:38: $finish called at 70 (1ns)
[Done] exit with code=0 in 0.298 seconds
```

D Flipflop:

```
module d_flipflop (
  input wire clk,
  input wire reset,
  input wire d,
  output reg q
);
  always @(posedge clk or posedge reset) begin
    if (reset)
      q <= 1'b0;
    else
      q <= d;
  end
endmodule
```

```
module d_flipflop_tb;
  reg clk;
  reg reset;
  reg d;
  wire q;
  d_flipflop uut (
    .clk(clk),
    .reset(reset),
    .d(d),
    .q(q)
  );
  initial begin
    clk = 0;
    reset = 1;
    d = 0;
    #10 reset = 0;
    #10 d = 1;
    #10 d = 0;
    #10 d = 1;
    #10 $finish;
  end
  always begin
    #5 clk = ~clk;
  end
  always @(posedge clk) begin
  $display("Time %0t: clk = %b, reset = %b, d = %b, q = %b", $time, clk, reset,
d, q);
  end
endmodule
```

OUTPUT:

```
[Running] lab9_2B.v
Time 5: clk = 1, reset = 1, d = 0, q = 0
Time 15: clk = 1, reset = 0, d = 0, q = 0
Time 25: clk = 1, reset = 0, d = 1, q = 0
Time 35: clk = 1, reset = 0, d = 0, q = 1
Time 45: clk = 1, reset = 0, d = 1, q = 0
lab9_2B.v:39: $finish called at 50 (1s)
[Done] exit with code=0 in 0.296 seconds
```

Task3: (i) Design D Flip-flop

(a) With synchronous reset

(b) With asynchronous reset and preset with reset having highest priority

(ii) Write a test-bench to check functionality of the module in 3(i).

a)

Code:

```
module flipflop(
    input d,clk,r,
    output reg q
);
always @(posedge clk or posedge r) begin
    if(r)
        q<=1'b0;
    else
        q<=d;
end
endmodule
module flipflop_tb;
  reg clk;
  reg r;
  reg d;
  wire q;

  flipflop uut (
    .d(d),
    .clk(clk),
    .r(r),
    .q(q)
  );

  initial begin
    clk = 0;
    forever #5 clk = ~clk;
  end
  initial begin
    r = 0;
    d = 0;
    #10 d = 1;
    #10 d = 0;
    #10 d = 1;
    #10 r = 1;
    #10 r = 0;
```

```
    #10 $finish;
  end

  always @(posedge clk) begin
    $display("Time %0t: clk = %b, r = %b, d = %b, q = %b", $time, clk, r, d, q);
  end
endmodule
```

OUTPUT:

```
[Running] lab9_3A.v
Time 5: clk = 1, r = 0, d = 0, q = x
Time 15: clk = 1, r = 0, d = 1, q = 0
Time 25: clk = 1, r = 0, d = 0, q = 1
Time 35: clk = 1, r = 0, d = 1, q = 0
Time 45: clk = 1, r = 1, d = 1, q = 0
Time 55: clk = 1, r = 0, d = 1, q = 0
lab9_3A.v:37: $finish called at 60 (1s)
[Done] exit with code=0 in 0.306 seconds
```

b) Code:

```
module flipflops(
    input d, clk, r ,pr,
    output reg q
);
    always @(posedge clk)
    begin
        if (r == 1'b1)
            q <= 1'b0;
        else if (pr == 1'b1)
            q <= 1'b1;
        else
            q <= d;
    end
endmodule
module flipflops_tb;
  reg clk;
  reg r;
  reg pr;
  reg d;
  wire q;
  flipflops uut (
    .d(d),
```

```
    .clk(clk),
    .r(r),
    .pr(pr),
    .q(q)
  );
  initial begin
    clk = 0;
    forever #5 clk = ~clk;
  end
  initial begin
    r = 0;
    pr = 0;
    d = 0;
    #10 d = 1;
    #10 d = 0;
    #10 pr = 1;
    #10 r = 1;
    #10 r = 0;
    #10 pr = 0;
    #10 $finish;
  end
  always @(posedge clk) begin
    $display("Time %0t: clk = %b, r = %b, pr = %b, d = %b, q = %b", $time, clk,
r, pr, d, q);
  end
endmodule
```

OUTPUT:

```
[Running] lab9_3B.v
Time 5: clk = 1, r = 0, pr = 0, d = 0, q = x
Time 15: clk = 1, r = 0, pr = 0, d = 1, q = 0
Time 25: clk = 1, r = 0, pr = 0, d = 0, q = 1
Time 35: clk = 1, r = 0, pr = 1, d = 0, q = 0
Time 45: clk = 1, r = 1, pr = 1, d = 0, q = 1
Time 55: clk = 1, r = 0, pr = 1, d = 0, q = 0
Time 65: clk = 1, r = 0, pr = 0, d = 0, q = 1
lab9_3B.v:44: $finish called at 70 (1s)
[Done] exit with code=0 in 0.238 seconds
```

4. For some application in Embedded design, a few clock pulses (clka, clkb, clkc) are needed as shown in Fig. 1. Generate these clock pulses using behavioral modeling (without using delay assignments).

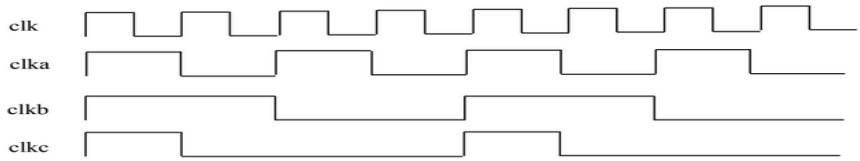


Fig. 1. Clock signals.

Code:

```
module clk_generator (
    input clk,
    output reg clka, clkb, clkc
);

Initial begin
    clka = 1'b0;
    clkb = 1'b0;
    clkc = 1'b0;
    $dumpfile("clk_generator.vcd");
    $dumpvars(0, clk_generator);
end
always @(posedge clk)
    clka <= ~clka;
always @(posedge clka)
    clkb <= ~clkb;
always @(posedge clkb)
    clkc <= 1'b1;
always @(negedge clka)
    clkc <= 1'b0;
endmodule

`timescale 1ns/1ns
module tb_clk_generator;
  reg clk;
  wire clka, clkb, clkc;
  clk_generator uut (
    .clk(clk),
    .clka(clka),
    .clkb(clkb),
    .clkc(clkc)
  );
  initial begin
    clk = 0;
    forever #5 clk = ~clk;
```

```
  end
  initial begin
    #10;
    $display("Initial values: clka=%b, clkb=%b, clkc=%b", clka, clkb, clkc);
    #20;
    $display("After 20 cycles: clka=%b, clkb=%b, clkc=%b", clka, clkb, clkc);
    #30;
    $display("After 30 cycles: clka=%b, clkb=%b, clkc=%b", clka, clkb, clkc);
    #40;
    $display("After 40 cycles: clka=%b, clkb=%b, clkc=%b", clka, clkb, clkc);
    #50;
    $display("After 50 cycles: clka=%b, clkb=%b, clkc=%b", clka, clkb, clkc);
    #10;
    $finish;
end
endmodule
```

OUTPUT:

```
[Running] lab9_4.v
VCD info: dumpfile clk_generator.vcd opened for output.
Initial values: clka=1, clkb=1, clkc=1
After 20 cycles: clka=1, clkb=0, clkc=0
After 30 cycles: clka=0, clkb=1, clkc=0
After 40 cycles: clka=0, clkb=1, clkc=0
After 50 cycles: clka=1, clkb=0, clkc=0
lab9_4.v:59: $finish called at 160 (1ns)
[Done] exit with code=0 in 0.237 seconds
```