Memo 4

Drexel University

To: Dr Christopher Peters
From: Mahad Faisal
Date: 02/15/2024
Re: Lab 3- Latches and Flip-Flops

Purpose

The purpose of this lab is to expand the understanding of Latches and Flip-Flops, the building blocks of memory. This is done through the simulation of a D-Latch and a D-Flip-Flop on EDAPlayground with Verilog, a hardware descriptive language, as well as building a D-Latch circuit using an Arduino MEGA 2560 kit as well as logic gate ICs.

Methodology

Before the Verilog simulations could be initiated, it was important to understand what D-Latches are, how they function and what their logic diagram looks like. A D-Latch is a type of circuit that stores memory. The D-Latch takes in two signals, a data signal (D) and an enable signal (EN) and outputs two variables Q and Q'. A D-Latch circuit is enable sensitive which means that the outputs, Q and Q', will only be affected by the data signal when the enable (EN) signal is set to HIGH or 1. If the enable signal is set to low the output signal(s) will remain the same as just before the enable signal changed and will only be affected by the data signal once the enable signal has been set to high again.

To better understand D-latches, both the behavioral and structural models of the D-latch circuit will be simulated using Verilog. In the behavioral model, equations are used as a tool to model the system, whereas in the structural model equations are replaced by the components of the D-latch as shown in **Fig 1.1**.

Since the first part of the project, 4A, will involve 2 models, the behavioral and structural, the design.sv file will contain two modules as seen in **Fig 2.1**. The module of the structural model has code that declares the inputs and outputs of the system, declares the wires N1, A1 and A and then it uses logic such as and/nor to set up the D-Latch. Unlike structural model, the behavioral model uses an equation inside of an always block to simulate the D-latch.

The testbench.sv file, seen in **Fig 2.2** is needed to run the simulation. After the variables, registers and wires have been declared, the two objects dls (structural) and dlb (behavioral) are instantiated. After that an initial block is created in which number of edge transitions made by the enable signal, basically how many times the enable signal changes, is set to 20. In the initial block the $monitor function is also used to display the variables and a for loop is used to determine a random shift time for the next EN and D start which are then delayed. This is used to produce non-periodic signals.

The following initial block created a .vcd dumpfile for the output(s) and dumps the variables into it. With this code both the structural and behavioral models are simulated,

The second part of the project, 4B, involves D Flip-Flops and their simulation. Like for the D-latch simulation, this will have both behavioral and structural models. The goal is to see that both models give the same output.

Because the D Flip-Flop is just two D Latches in series, the Verilog code for the flip-flop is very similar to that of the D-Latch, and this is seen in **Fig. 2.3.** The timescale function is initially used to specify the length and precision o the simulation the structural model for the D latch that makes up the flip flop is created.

Just as in the previous part of the lab, the structural model uses logic commands/functions such as 'and' to set up the d latch. The following module addresses the output of the flip flop and assigns the variables to wires to drive the output. In tis module the code checks for a positive clock edge and because of this when a positive clock edge is recognized the output, Q, is set to D.

The behavioral model uses an equation in an always block to set the system. The following testbench is also very similar to that of the d-latch code seen before. Once the dut1 and dut2 variables are created for the structural and behavioral outputs respectively and initial begin block similar to the one in the previous example is created, a vcd dumpfile is made and variables are dumped into it. Before the code is finished, a line that generates the clock signal with a time period of 4 steps is made and the initial clock value is set to 0. This code perfectly simulates the D Flip-flop as required.

The final part of the Verilog section of this project involved modifying the testbench file from project 4B to accommodate 1 D-Latch and 1 D Flip-Flop and their respective outputs, so that the functions of both can be compared. In this code [**Fig 2.4**] only the behavioral model is used for the flip flop and only the structural for the latch as the aim is to compare the different types of circuits not different ways to implement them. In this code two object were instantiated, the structural model of the D latch and the behavioral model of the flip flop in contrast to the reference code.

Once the modifications were made to the code, it was run using Icarus Verilog 0.9.7 and EPwave on EDAPlayground.

The final part of this lab was to create a physical circuit, **Fig 3**, using the Arduino MEGA 2560 kit and logic ICs that stored 1 bit of memory which is a D latch. The components used to build this circuit were:

- 1) Large breadboard
- (1) Arduino Mega 2560
- (1) 74HC04 Hex inverter package
- (1) 74HC08 AND gate package
- (1) 74HC02 NOR gate package
- (2) Buttons
- (2) 5 kΩ resistors

- (2) 1 kΩ resistors
- Jumper wires

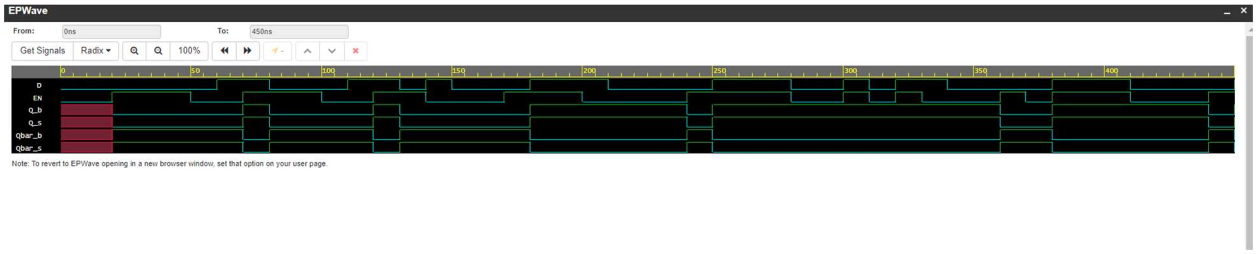
To start building this circuit, buttons were added to the large breadboard. For power the top left of the buttons were connected to the top power rail and for grounding and current control 5 kiloohm resistors were connected from the bottom right pin of the buttons to the bottom blue ground rail.

After that the three logic gates 74HC04, 08 and 02 were placed onto the board in that order. The VCC pins for these logic gate ICs were connected to the top power rail and the GND pins were connected to the bottom GND rail. Once those connections were verified the two LEDs were added with their respective 1 kilo-ohm resistors connected to the bottom blue ground rail.

To finish off, the wring for the logic gates for placed and verified and Arduino board was connected to the breadboard via wires. After powering the Arduino via the wall outlet pin, the Arduino code was uploaded to the board and the different combinations of inputs on the buttons were tested and the results recorded via the serial plotter built into the Arduino IDE. This was the project in its entirety.

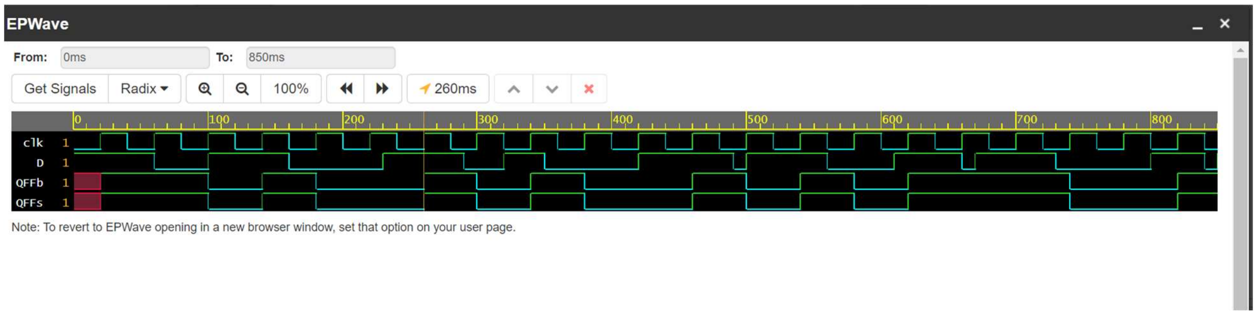
Results

Project 4A results


EPWave
From: 0ns
To: 450ns
Get Signals
Radix
100%
0
50
100
150
200
250
300
350
400
D
EN
Q_b
Q_s
Qbar_b
Qbar_s
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

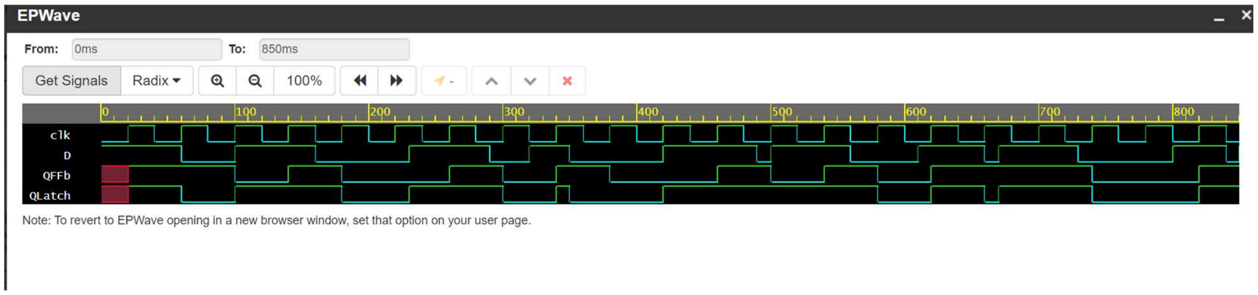
Pictured above is the EPwave output of project 4A. The desired output was achieved according to the output as both the structural and behavioral outputs were the same for outputs Q and Q'.

Project 4B results

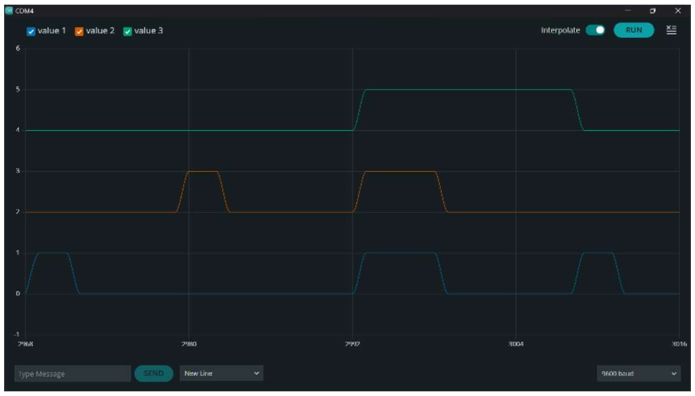


Pictured above is the EPwave output for project 4B. The goal of this part was to get the right output for a D Flip-Flop and the waveform above shows that the goal was achieved because the output for both the behavioral and structural model is edge triggered so it only changes when the enable or clock/clk value changes from low to high and high to low as it should.

Verilog 4C results



According to logic theory, flip-flops are edge triggered so its output changes when the control/clk signal goes from high to low and latches are level triggered so its output change whenever the input changes unaffected by the clk signal. This phenomenon can clearly be seen in the waveform output seen above. The latch clearly responds to changes in the input, D, and the flip-flop responds to rising edges in the clock signal.



The goal of the hardware lab was to visualize the correct set of data (inputs and outputs) for different combinations of inputs (e.g. only left pressed, both pressed then released etc.) in the serial plotter of the Arduino IDE which is pictured above. This serial pot shows that the D latch was built successfully. Value 3 is the output, value 2 is the data signal and value 1 is the clock signal. The output signal doesn’t change when only one of the data and clock signal are energized at the beginning, but when both are activated simultaneously, the output returns a HIGH value. Even after the data and clock signal are de-energized the output signal continues to return a HIGH value because the function of the d latch is to latch onto the last given value and store it, and it only resets to low once the clock signal is energized at the end.

Appendix

Figure 5.1 A D-Latch circuit

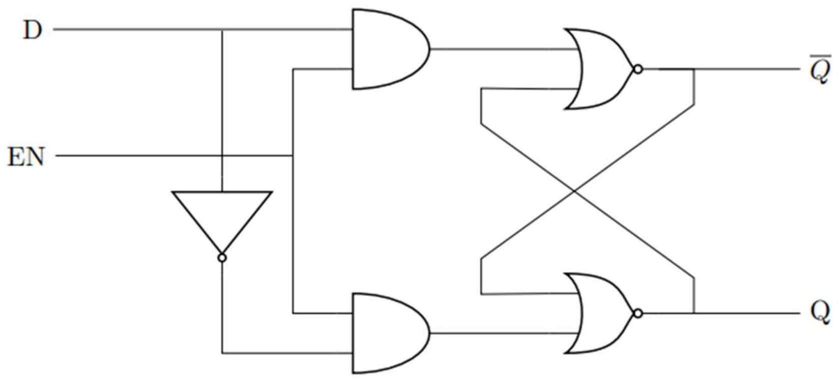
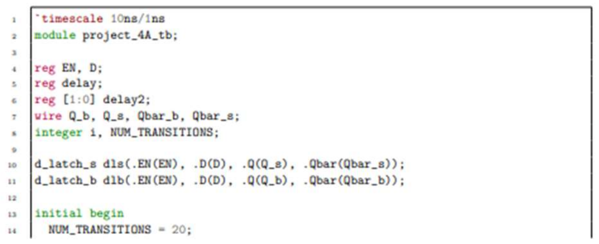


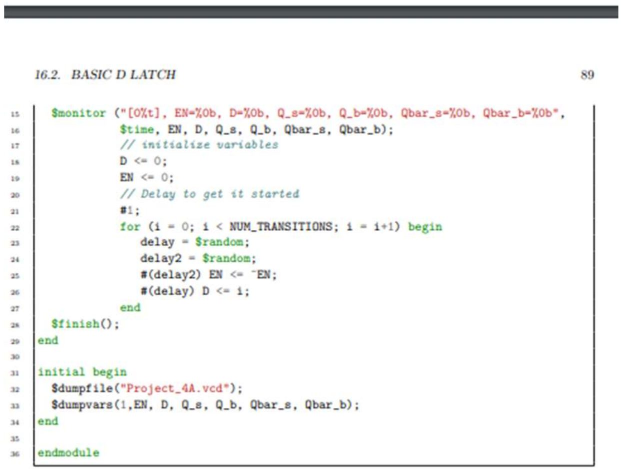
Figure 6.1 The design.sv module for the d-latch

```
// Structural model of D latch
module d_latch_s(input EN, input D, output Q, output Qbar);

wire N1, A1, A2;

not(N1,D);
and(A1, D, EN);
and(A2, N1, EN);
nor(Qbar, A1, Q);
nor(Q, A2, Qbar);

endmodule

// Behavioral model of D Latch
module d_latch_b(input EN, input D, output reg Q, output reg Qbar);

always @(EN or D)
   if(EN) begin
      Q <= D;
      Qbar <= ~D;
    end
endmodule
```

Figure 2.2 testbench.sv file for d-latch simulation



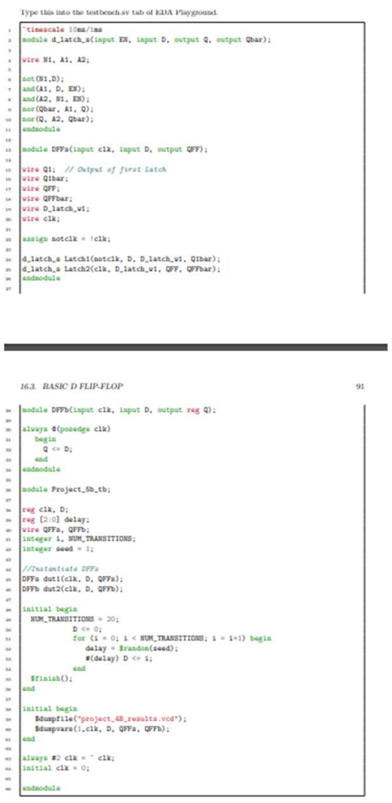
```
`timescale 10ns/1ns
module project_4A_tb;

reg EN, D;
reg delay;
reg [1:0] delay2;
wire Q_b, Q_s, Qbar_b, Qbar_s;
integer i, NUM_TRANSITIONS;

d_latch_s dls(.EN(EN), .D(D), .Q(Q_s), .Qbar(Qbar_s));
d_latch_b dlb(.EN(EN), .D(D), .Q(Q_b), .Qbar(Qbar_b));

initial begin
  NUM_TRANSITIONS = 20;
```



16.2. BASIC D LATCH 89

```
  $monitor ("[0%t], EN=%0b, D=%0b, Q_s=%0b, Q_b=%0b, Qbar_s=%0b, Qbar_b=%0b",
            $time, EN, D, Q_s, Q_b, Qbar_s, Qbar_b);
            // initialize variables
            D <= 0;
            EN <= 0;
            // Delay to get it started
            #1;
            for (i = 0; i < NUM_TRANSITIONS; i = i+1) begin
               delay = $random;
               delay2 = $random;
               #(delay2) EN <= ~EN;
               #(delay) D <= i;
            end
  $finish();
end

initial begin
  $dumpfile("Project_4A.vcd");
  $dumpvars(1,EN, D, Q_s, Q_b, Qbar_s, Qbar_b);
end

endmodule
```

Figure 2.3 testbench.sv code for flip flop



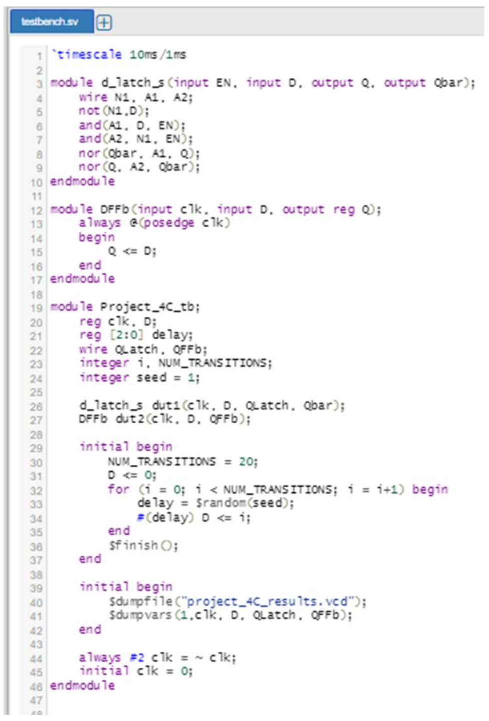
Type this into the testbench.sv tab of EDA Playground.

```
`timescale 10ms/1ms
module d_latch_s(input EN, input D, output Q, output Qbar);

wire N1, A1, A2;

not(N1,D);
and(A1, D, EN);
and(A2, N1, EN);
nor(Qbar, A1, Q);
nor(Q, A2, Qbar);
endmodule

module DFFs(input clk, input D, output QFF);

wire Q1;  // Output of first Latch
wire Q1bar;
wire QFF;
wire QFFbar;
wire D_latch_w1;
wire clk;

assign notclk = !clk;

d_latch_s Latch1(notclk, D, D_latch_w1, Q1bar);
d_latch_s Latch2(clk, D_latch_w1, QFF, QFFbar);
endmodule

```

16.3. BASIC D FLIP-FLOP 91

```
module DFFb(input clk, input D, output reg Q);

always @(posedge clk)
   begin
     Q <= D;
   end
endmodule

module Project_5b_tb;

reg clk, D;
reg [2:0] delay;
wire QFFs, QFFb;
integer i, NUM_TRANSITIONS;
integer seed = 1;

//Instantiate DFFs
DFFs dut1(clk, D, QFFs);
DFFb dut2(clk, D, QFFb);

initial begin
  NUM_TRANSITIONS = 20;
            D <= 0;
            for (i = 0; i < NUM_TRANSITIONS; i = i+1) begin
               delay = $random(seed);
               #(delay) D <= i;
            end
  $finish();
end

initial begin
   $dumpfile("project_4B_results.vcd");
   $dumpvars(1,clk, D, QFFs, QFFb);
end

always #2 clk = ~ clk;
initial clk = 0;

endmodule
```

Figure 2.4 testbench



```
`timescale 10ms/1ms

module d_latch_s(input EN, input D, output Q, output Qbar);
    wire N1, A1, A2;
    not(N1,D);
    and(A1, D, EN);
    and(A2, N1, EN);
    nor(Qbar, A1, Q);
    nor(Q, A2, Qbar);
endmodule

module DFFb(input clk, input D, output reg Q);
    always @(posedge clk)
    begin
        Q <= D;
    end
endmodule

module Project_4C_tb;
    reg clk, D;
    reg [2:0] delay;
    wire QLatch, QFFb;
    integer i, NUM_TRANSITIONS;
    integer seed = 1;

    d_latch_s dut1(clk, D, QLatch, Qbar);
    DFFb dut2(clk, D, QFFb);

    initial begin
        NUM_TRANSITIONS = 20;
        D <= 0;
        for (i = 0; i < NUM_TRANSITIONS; i = i+1) begin
            delay = $random(seed);
            #(delay) D <= i;
        end
        $finish();
    end

    initial begin
        $dumpfile("project_4C_results.vcd");
        $dumpvars(1,clk, D, QLatch, QFFb);
    end

    always #2 clk = ~ clk;
    initial clk = 0;
endmodule

```

Figure 7 Finished Arduino circuit