Memo 3

Drexel University

To: Dr Christopher Peters
From: Mahad Faisal
Date: 02/09/2024
Re: Lab 3- Decoders

Purpose

The aim of 'Lab 3-Decoders' assignment was to recreate/simulate the lights on the Knight Rider Two Thousand car from the 80s Knight Rider show, building a mini K.I.T.T. This was done by, first, modelling the behavior of a real decoder, in this case the SN74HC138 IC, then simulating that behavior using Verilog on EDAPlayground and finally building it using an ARDUINO MEGA 2560 kit.

Methodology

The first part of this project was to model the behavior of the SN74HC138 decoder before implementing it. This was done by looking at the schematics of the decoder, **Fig 1.1**. As seen on the image, there are multiple pins, groups of pins and variables that must be understood and characterized.

The VCC pin seen on the diagram is a power pin, so if there is no power connected to this pin, the output pins cannot be set to '1' or HIGH. The GND pin pictured on the bottom-left hand side of the image is the grounding pin and if the grounding pin isn't connected to a ground reference point the decoder will likely not initialize and damage to the equipment may occur. Pins 1, 2 and 3 on the schematic are variables A, B, and C and this is where the values of the inputs are connected and driven through the logic operations. The order of these pins, in which A is at the top and C at the bottom, is very important to the system. This is because this arrangement assigns A as the Least Significant Bit and C as the Most Significant Bit, which will determine the order of the numbers in the output and to achieve the desired output and KITT effect as discussed in the purpose.

Pins 4, 5, and 6 are enable pins for the variables discussed above. Pin G'1A and pin G'2A set the values of A and B to active LOW, and pin G1 sets the value of C as active HIGH. These values will be stored as registers as they hold values. Finally, pins Y0-Y7 are the output pins and are represented as a wire Y with vector [7:0].

This schematic and its explanation was used to model the behavior of the decoder and after this the Verilog simulation was created.
Before the testbench file could be created, the design.sv source code file was written as seen in Fig **2. 1**. Line 1-3 start the module named 'SN74HC138' and declare the inputs and outputs of the system that were discussed above. Lines 4-6 declare some variables as registers; Line 4 regs have 1 bit, the Line 5 reg is 8 bits and the Line 6 reg has 3 bits. Line 8 and Line 34 start and end an always block which will be executed whenever one of any of the variables changes. The normal_ops line 9 sets the value for normal operations to HIGH or 1 when all variables except G1A_bar or G2A_bar are set to true o high. The following all_LOW line sets it to true whenever VCC or GND is false so whenever there is no power input or whenever there is no ground connection as a failsafe to protect from circuit damage.

Conversely, the all_HIGH line sets itself to true when both VCC and GND are HIGH and when either G2A_bar or G2B_bar or **not** G1 is HIGH. Line 14 then executes a bitwise concatenation of the variables C, B, and A in that order to the variable **in** to make it easier to manipulate the **out** variable. The if block in Lines 15-29 sets the initial output to all 1s (as the decoder's outputs are all ACTIVE LOW) and then depending on the value of **in** one of the 1s will be replaced with a 0. The following lines, 30-33, create the condition in which when in the ALL HIGH mode, each bit is set to HIGH and when in the ALL LOW mode every bit of the output is set to LOW. The final two lines then end the always block and the module respectively.

A reference testbench file was provided for an 8 bit K.I.T.T system [**Fig 2.21**]. Line 1 of this testbench.sv file simply starts the module 'SN74HC138_tb'. Lines 2 and 5 then declare the variables. The wire '**out'** is assigned with a vector of [7:0], therefore of length 8 and register '**in**' is assigned with a vector of [2:0] so a length of 3. Line 5 declares an integer '**i**' which will be integral in the system later on for going through all the different iterations of the variables.

Line 7 instantiates an object 'dut' with a type 'SN74HC138' and the figures inside the brackets are the port connections in the system. After all the preliminary code is finalized 'initial begin' in line 9 is used to set the variables within the system e.g. VCC and GND =1 as the system must have power and must be grounded to work and the rest of the variables are set to the values from the normal_ops line in the design.sv file so that the system operates in normal mode.

The '$monitor' function in the initial begin block is used to display the values specified inside the function (inside the brackets) in the same order used inside the function. Lines 17-21 use the integer '**i**' assigned at the beginning of the code; they do this by creating the values of inputs C, B, and A by increasing i and then by assigning those values to their respective variables. In each of the iterations caused by this, at least one of variables C, B and A are changed and this activates the always block inside the design.sv file. The second iteration block creates the values of C, B and A.

Line 27 then terminates the initial begin block and then lines 30-33 creates the vcd dumpfile for the waveform. Finally, the entire module is terminated in line 34 with the command 'endmodule'

This program was run on EDAPlayground and the waveform was displayed using the built in EPWave program.

As mentioned previously, that testbench file produces an 8 bit K.I.T.T output and a 16 bit system is required for this project. To create a 16 bit output with this setup, only the testbench.sv file had to be modified as the design.sv file was compatible with both renditions of the decoder.

The version of code for the 16 bit K.I.T.T system can be seen in **Fig 2.22.** The first change that made was to assign the wire to two out values, out1 and out2, as making this 16 bit decoder involves using 3x8 decoders and then modifying the vector length accordingly to 8, so the new vector was [7:0]. Another necessary change was to add 'G1A' as a register as the second dut object, that was subsequently added to the file, needed a different variable so that the output does not repeat. The 'in' assigned to the reg was also replaced by 'out' as the out holds the concatenation of out1 and out2 and that is better name.

As mentioned in the previous paragraph after the variables were set, a dut2 object was added that was the same type and held the same values as dut1 but replaced G1 by G1A. In order to denote that G1A had to be the opposite of G1, an always block was created (Lines 10-12). In the initial begin block of the 8-bit system, G1 had an assigned value but in this rendition the value of G1 should alternate between HIGH and LOW so it was added to the for loop as the Most Significant Bit to get the desired visual effect.

At this point in the process there was a major hurdle that was seemingly unresolvable and it took hours of debugging to fix. The output kept producing an output with all values set to LOW and that ruined the structure of the outputs and the desired effect wasn't achieved and because the vector length of 16 the final desired output was cut out because of this unwanted piece of data. It was then determined that the reason why the output was showing that was because 'out' was not given an initial value so it was then initialized with a value of 0 in order to get the right structure and the right outputs.

Because this the vector length in this system is different, 'i<8' was replaced by 'i<16' to account for the new length of 16 in both for loops. Then 'out' 'was also added in the $dumpvars line so that the out value was added to the vcd file. Adding the always block before ending the module was also necessary because without that the output would be all 1s with one 0 but the project requires all 0s and one moving 1 and this block both concatenates the out1 and out2 wires and also gives the inverse of those values. Running this version of the code gave the correct and desired output.

The second and final part of the project was to create the K.I.T.T. light system using hardware. The hardware used in this system was:

- Large breadboard
- Arduino Mega 2560
- 74HC138 3-to-8 decoder
- 74HC04 hex inverter
- (8) LEDs
- (8) 1 kΩ resistors

The circuit pictured in **Fig 3.1** involves a lot of carefully placed wiring and this was important because even a single misplaced wire could cause serious damage to the components.

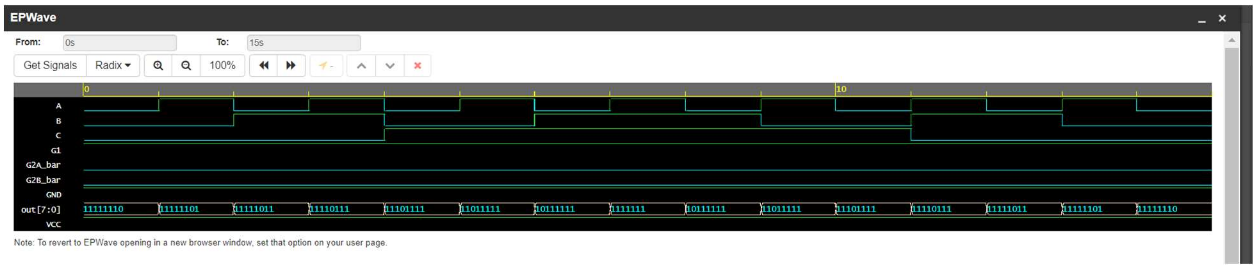
First the 74HC138 3-to-8 decoder and the two 74HC04 NOT gates were placed on the breadboard and then a common power wire was connected between the top positive rail and the bottom positive rail. Individual positive connections were also made by connecting the VCC pins on the packages to the top positive rail. Then grounding wires were added to all 3 packages by connecting their GND pins to the bottom negative rail. Because variable G1 changes and requires power pin 6 on the 74HC138 (G1) was connected to the bottom positive rail.

After that the 8 LEDs were added to the circuit and then a 1k ohm resistor was connected between the LED and the negative bottom rail of the breadboard in order to limit the current flowing and prevent burn-out. Then the LEDs were connected with wires to the required/correct inputs on the logic gate e.g. Decoder pin Y0 to pin 6A of the first not gate and then to the first LED to get the right structure of outputs.

Once all of the wiring for the LED was completed the Arduino MEGA 2560 was connected to the board and the correct logic gate inputs, the 5V power pin to the positive bottom rail and the GND pin to the negative bottom rail. The circuit, once all connections were verified, was then powered with the wall power brick and connected to a PC via USB and the Arduino code for this simulation was uploaded and the simulation executed. As expected the all but one light was on at a time and the HIGH value travelled up and down the row of lights meaning that the project was successful.

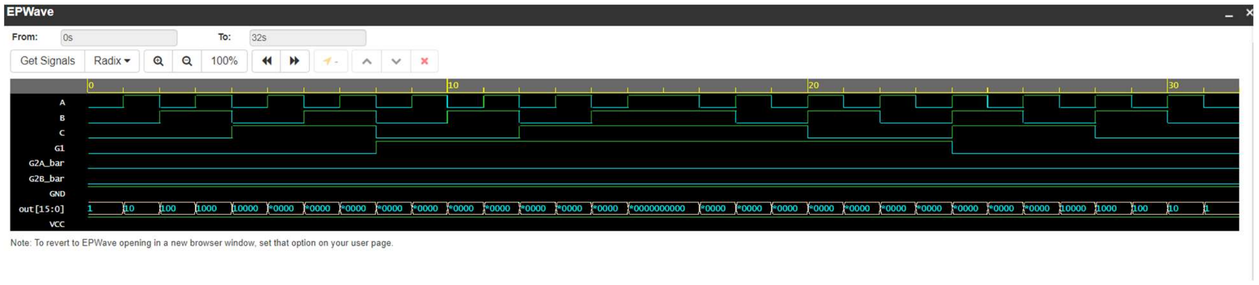
Results

Project 3A results

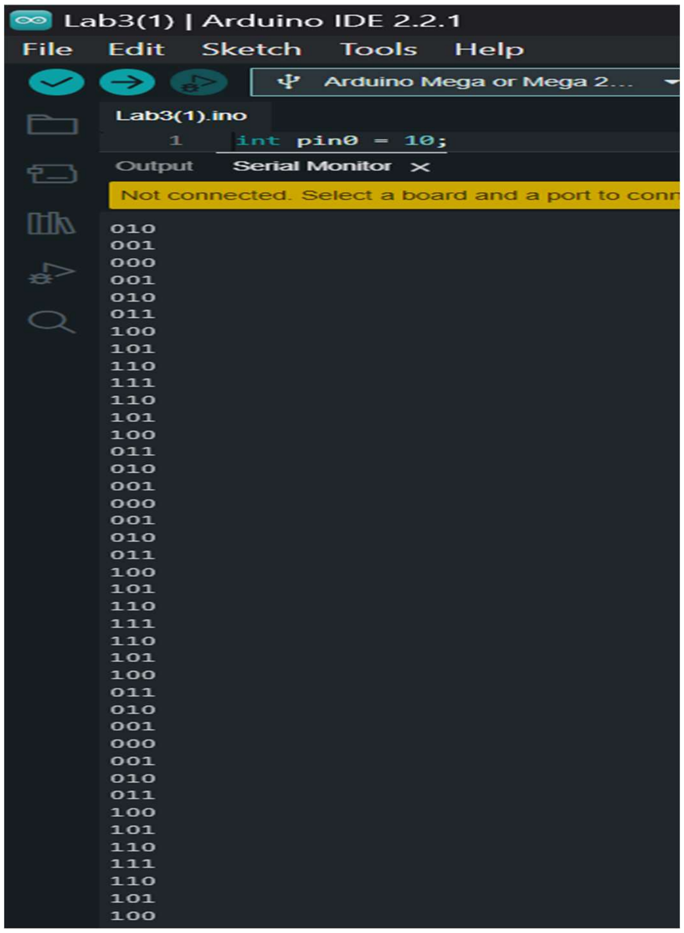


Note: To revert to EPWave opening in a new browser window, set that option on your user page.

PROJECT 3B RESULTS



Both of these waveforms show that the desired results were obtained in both simulations. The results of 3A show that an 8bit output was achieved with all outputs set to HIGH except for one at a time. The time period of the 3A waveform also confirms the accuracy of the results as it goes from 0 to 15s which means that the correct vector length of 16 was utilized. Similarly in the 3B results, although those are seen clearer on the log in **Figure 2.3**, show that the correct output was generated. This project was successful and it made a substantial impact on my Verilog knowledge and skills.



```
int pin0 = 10;
```

```
Not connected. Select a board and a port to conn
```

```
010
001
000
001
010
011
100
101
110
111
110
101
100
011
010
001
000
001
010
011
100
101
110
111
110
101
100
011
010
001
000
001
010
011
100
101
110
111
110
101
100
```

With the Arduino simulation a truth table was also developed and while it looped which it should according to the program which is needed for the desired effect, it follows the logic operations and is valid.

Appendix

Figure 1.1

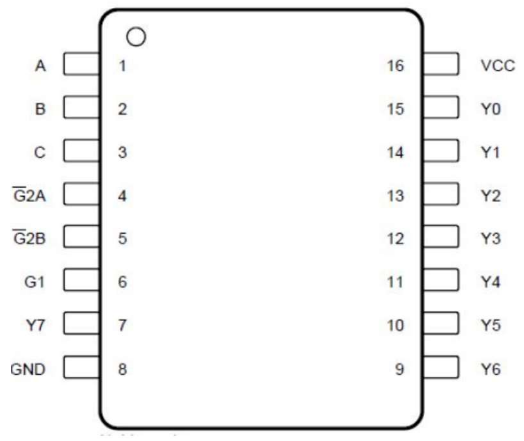
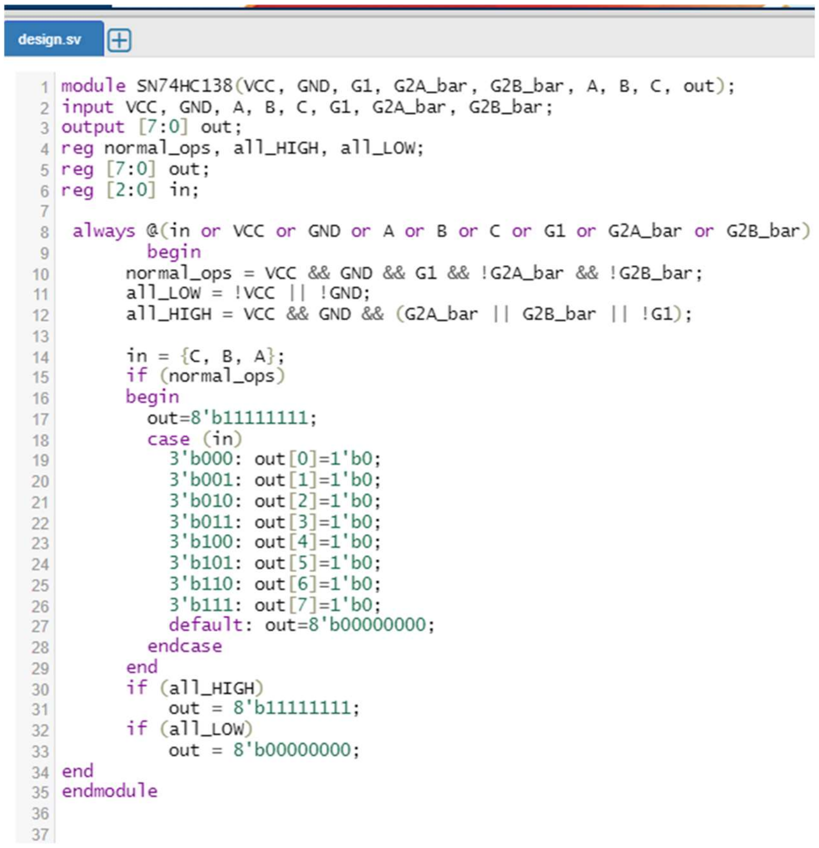
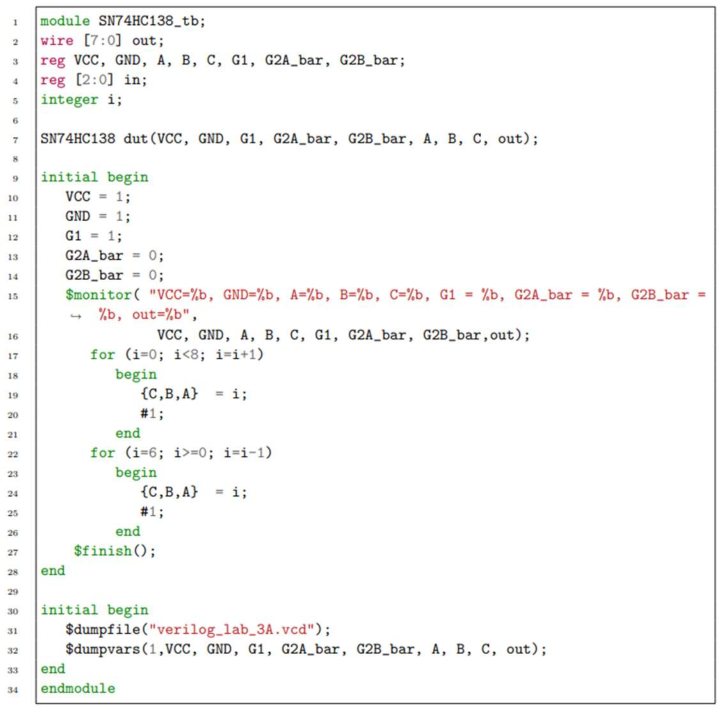


Figure 2.1



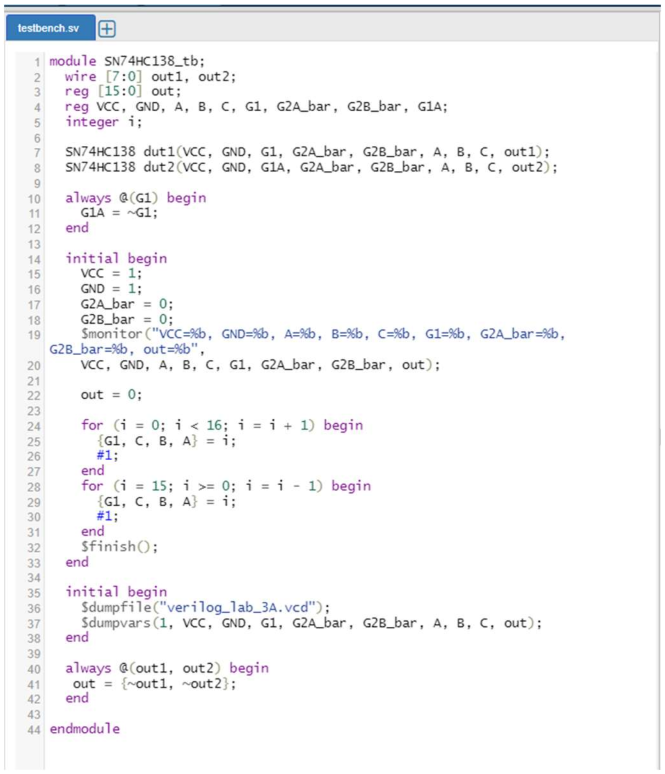
```
module SN74HC138(VCC, GND, G1, G2A_bar, G2B_bar, A, B, C, out);
input VCC, GND, A, B, C, G1, G2A_bar, G2B_bar;
output [7:0] out;
reg normal_ops, all_HIGH, all_LOW;
reg [7:0] out;
reg [2:0] in;

 always @(in or VCC or GND or A or B or C or G1 or G2A_bar or G2B_bar)
        begin
      normal_ops = VCC && GND && G1 && !G2A_bar && !G2B_bar;
      all_LOW = !VCC || !GND;
      all_HIGH = VCC && GND && (G2A_bar || G2B_bar || !G1);

      in = {C, B, A};
      if (normal_ops)
      begin
        out=8'b11111111;
        case (in)
          3'b000: out[0]=1'b0;
          3'b001: out[1]=1'b0;
          3'b010: out[2]=1'b0;
          3'b011: out[3]=1'b0;
          3'b100: out[4]=1'b0;
          3'b101: out[5]=1'b0;
          3'b110: out[6]=1'b0;
          3'b111: out[7]=1'b0;
          default: out=8'b00000000;
        endcase
      end
      if (all_HIGH)
          out = 8'b11111111;
      if (all_LOW)
          out = 8'b00000000;
end
endmodule
```

Figure 2.21



```
module SN74HC138_tb;
wire [7:0] out;
reg VCC, GND, A, B, C, G1, G2A_bar, G2B_bar;
reg [2:0] in;
integer i;

SN74HC138 dut(VCC, GND, G1, G2A_bar, G2B_bar, A, B, C, out);

initial begin
   VCC = 1;
   GND = 1;
   G1 = 1;
   G2A_bar = 0;
   G2B_bar = 0;
   $monitor( "VCC=%b, GND=%b, A=%b, B=%b, C=%b, G1 = %b, G2A_bar = %b, G2B_bar =
   ↪  %b, out=%b",
               VCC, GND, A, B, C, G1, G2A_bar, G2B_bar,out);
      for (i=0; i<8; i=i+1)
         begin
            {C,B,A}  = i;
            #1;
         end
      for (i=6; i>=0; i=i-1)
         begin
            {C,B,A}  = i;
            #1;
         end
    $finish();
end

initial begin
   $dumpfile("verilog_lab_3A.vcd");
   $dumpvars(1,VCC, GND, G1, G2A_bar, G2B_bar, A, B, C, out);
end
endmodule
```

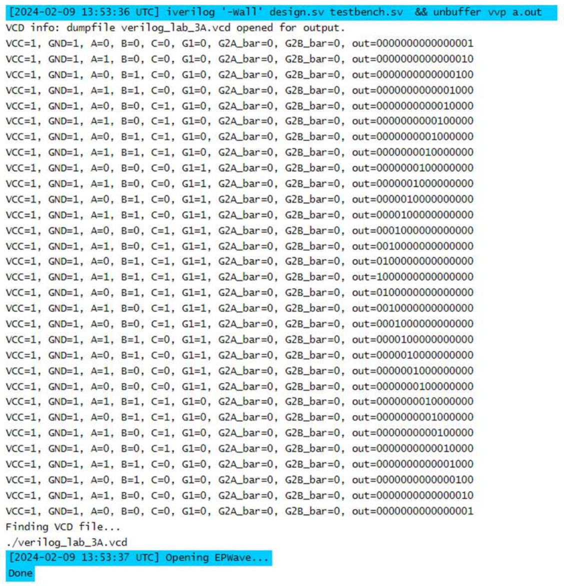
Figure 2.22



testbench.sv

```
module SN74HC138_tb;
  wire [7:0] out1, out2;
  reg [15:0] out;
  reg VCC, GND, A, B, C, G1, G2A_bar, G2B_bar, G1A;
  integer i;

  SN74HC138 dut1(VCC, GND, G1, G2A_bar, G2B_bar, A, B, C, out1);
  SN74HC138 dut2(VCC, GND, G1A, G2A_bar, G2B_bar, A, B, C, out2);

  always @(G1) begin
    G1A = ~G1;
  end

  initial begin
    VCC = 1;
    GND = 1;
    G2A_bar = 0;
    G2B_bar = 0;
    $monitor("VCC=%b, GND=%b, A=%b, B=%b, C=%b, G1=%b, G2A_bar=%b,
G2B_bar=%b, out=%b",
    VCC, GND, A, B, C, G1, G2A_bar, G2B_bar, out);

    out = 0;

    for (i = 0; i < 16; i = i + 1) begin
      {G1, C, B, A} = i;
      #1;
    end
    for (i = 15; i >= 0; i = i - 1) begin
      {G1, C, B, A} = i;
      #1;
    end
    $finish();
  end

  initial begin
    $dumpfile("verilog_lab_3A.vcd");
    $dumpvars(1, VCC, GND, G1, G2A_bar, G2B_bar, A, B, C, out);
  end

  always @(out1, out2) begin
   out = {~out1, ~out2};
  end

endmodule
```

Figure 2.3



```
[2024-02-09 13:53:36 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile verilog_lab_3A.vcd opened for output.
VCC=1, GND=1, A=0, B=0, C=0, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000000001
VCC=1, GND=1, A=1, B=0, C=0, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000000010
VCC=1, GND=1, A=0, B=1, C=0, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000000100
VCC=1, GND=1, A=1, B=1, C=0, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000001000
VCC=1, GND=1, A=0, B=0, C=1, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000010000
VCC=1, GND=1, A=1, B=0, C=1, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000100000
VCC=1, GND=1, A=0, B=1, C=1, G1=0, G2A_bar=0, G2B_bar=0, out=0000000001000000
VCC=1, GND=1, A=1, B=1, C=1, G1=0, G2A_bar=0, G2B_bar=0, out=0000000010000000
VCC=1, GND=1, A=0, B=0, C=0, G1=1, G2A_bar=0, G2B_bar=0, out=0000000100000000
VCC=1, GND=1, A=1, B=0, C=0, G1=1, G2A_bar=0, G2B_bar=0, out=0000001000000000
VCC=1, GND=1, A=0, B=1, C=0, G1=1, G2A_bar=0, G2B_bar=0, out=0000010000000000
VCC=1, GND=1, A=1, B=1, C=0, G1=1, G2A_bar=0, G2B_bar=0, out=0000100000000000
VCC=1, GND=1, A=0, B=0, C=1, G1=1, G2A_bar=0, G2B_bar=0, out=0001000000000000
VCC=1, GND=1, A=1, B=0, C=1, G1=1, G2A_bar=0, G2B_bar=0, out=0010000000000000
VCC=1, GND=1, A=0, B=1, C=1, G1=1, G2A_bar=0, G2B_bar=0, out=0100000000000000
VCC=1, GND=1, A=1, B=1, C=1, G1=1, G2A_bar=0, G2B_bar=0, out=1000000000000000
VCC=1, GND=1, A=0, B=1, C=1, G1=1, G2A_bar=0, G2B_bar=0, out=0100000000000000
VCC=1, GND=1, A=1, B=0, C=1, G1=1, G2A_bar=0, G2B_bar=0, out=0010000000000000
VCC=1, GND=1, A=0, B=0, C=1, G1=1, G2A_bar=0, G2B_bar=0, out=0001000000000000
VCC=1, GND=1, A=1, B=1, C=0, G1=1, G2A_bar=0, G2B_bar=0, out=0000100000000000
VCC=1, GND=1, A=0, B=1, C=0, G1=1, G2A_bar=0, G2B_bar=0, out=0000010000000000
VCC=1, GND=1, A=1, B=0, C=0, G1=1, G2A_bar=0, G2B_bar=0, out=0000001000000000
VCC=1, GND=1, A=0, B=0, C=0, G1=1, G2A_bar=0, G2B_bar=0, out=0000000100000000
VCC=1, GND=1, A=1, B=1, C=1, G1=0, G2A_bar=0, G2B_bar=0, out=0000000010000000
VCC=1, GND=1, A=0, B=1, C=1, G1=0, G2A_bar=0, G2B_bar=0, out=0000000001000000
VCC=1, GND=1, A=1, B=0, C=1, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000100000
VCC=1, GND=1, A=0, B=0, C=1, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000010000
VCC=1, GND=1, A=1, B=1, C=0, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000001000
VCC=1, GND=1, A=0, B=1, C=0, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000000100
VCC=1, GND=1, A=1, B=0, C=0, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000000010
VCC=1, GND=1, A=0, B=0, C=0, G1=0, G2A_bar=0, G2B_bar=0, out=0000000000000001
Finding VCD file...
./verilog_lab_3A.vcd
[2024-02-09 13:53:37 UTC] Opening EPWave...
Done
```

Figure 3.1