4 IN ONE COUNTER

ABSTRACT

The project is an 4-in-One Counter implemented using structural Verilog coding, incorporating decoders and AND gates for counter access and reset functionalities. Developed as part of an HDL course project at IIIT Nagpur under the guidanceof Mrs. Sushmita Dandeliya, the counter module integrates various counter modes, including up counters, down counters, Mod10 up counter and Mod5 down counter. The project demonstrates the versatility and flexibility of the counter module, allowing users to select and reset specific counters based on their requirements. By utilizing structural Verilog coding techniques, the design achieves a modular and organized representation of the counter components, making it readable and maintainable.The implementation of the project utilizes Xilinx FPGA for its rapid prototyping, hardware synthesis, and scalability capabilities, enabling real-time testing andvalidation of the counter functionalities.

INTRODUCTION

Counters are fundamental components in digital circuit design, enabling thecounting and tracking of events or signals. The 4-in-one Counter project aims to provide a comprehensive solution by integrating various counter modes into a single module. This project was undertaken as part of an HDL course project at IIIT Nagpur, guided by Mrs. Sushmita Dandeliya.

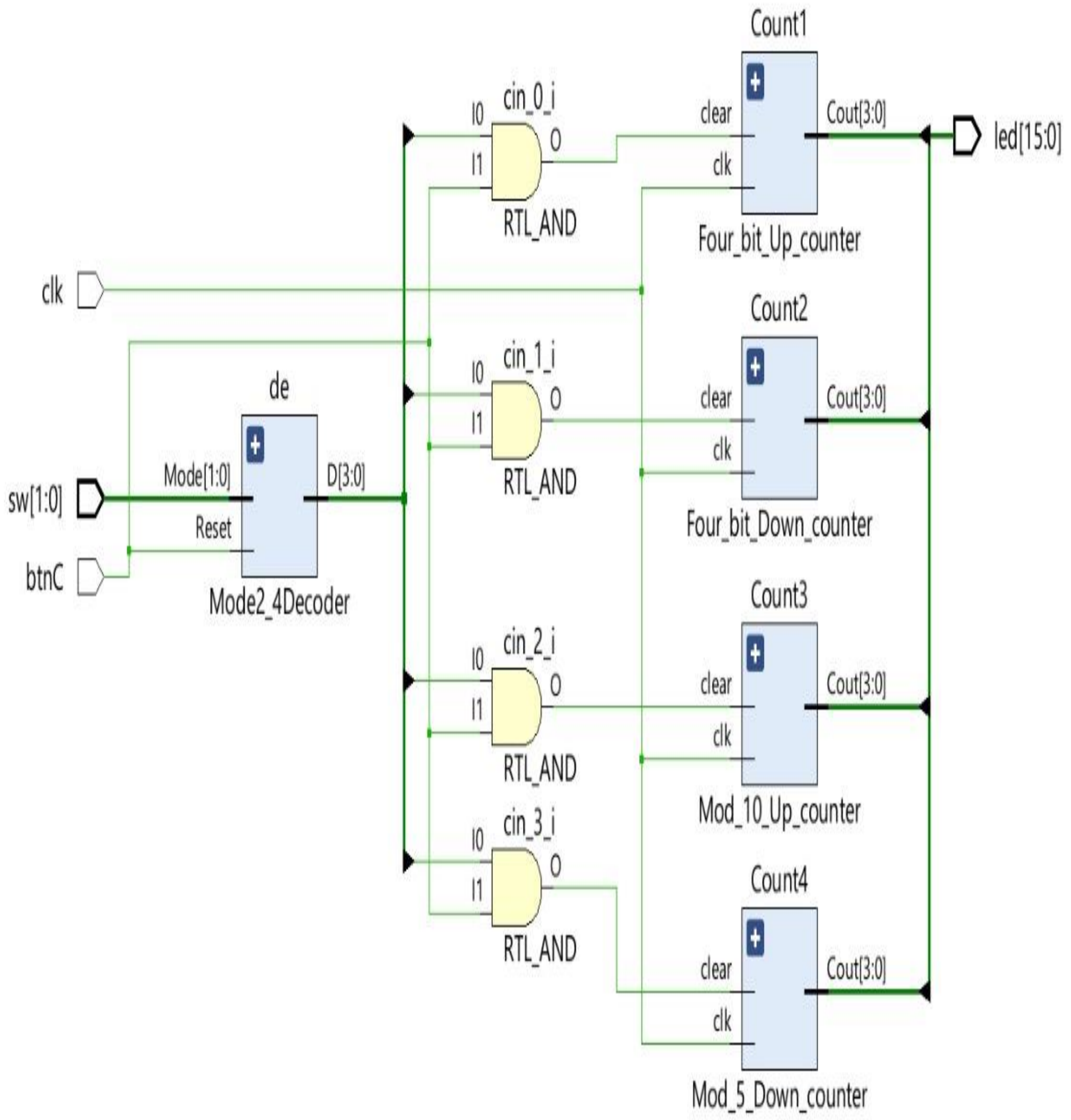
The 4-in-one Counter module in- corporates different counter modes suchas up counters, down counters, Mod10 up counters and Mod5 down counters. These counter modes offer versatile functionality for counting applications, accommodating different counting directions, modulo operations, and sequence patterns.

The project adopts structural Verilog coding, which enables the hierarchical composition of modules using gate level primitives. This coding style results in a modular and organized representation of the counter components, enhancing readability and maintain ability. Structural Verilog coding facilitates the seamless integration of decoders, AND gates, and counter modules, forming a coherent and functional 4-in-one Counter design.

For implementation, the project utilizes the Xilinx FPGA platform. The choice of Xilinx FPGA offers advantages such as rapid prototyping, hardware synthesis, and scalability. Thisplatform enables efficient synthesis of the 4-in-one Counter design onto the FPGA, facilitating real time testing and validation of the counter functionalities.

In summary, the 4-in-one Counter project serves as an educational tool, demonstrating the integra-tion of multiple counter modes using structural Verilog coding. By incorporating decoders and AND gates, the project enables counter access and reset operations, enhancing the versatility and functionality of the counter mod-ule. The project’s contributions lie in its practical application of digital circuit design concepts and its potential for further exploration and enhancements in the field of counters and related applications.

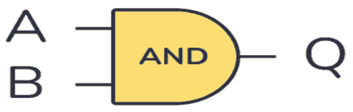
CIRCUIT DIAGRAM



DIGITAL CIRCUITS

AND Gate:

A 2-input AND gate is a basic digital logic gate that takes two binary inputs (A and B) and produces an output (Y) based on the logical AND operation between the inputs. The output ofthe AND gate is true (logic high) only when both input A and input B are true(logic high).
2-input AND gates are fundamental building blocks in digital logic circuits. They are used extensively in various applications, including arithmetic operations, data manipulation, signal processing, and control systems. They play a crucial role in performing logical operations and combining multiple signals in digital systems.

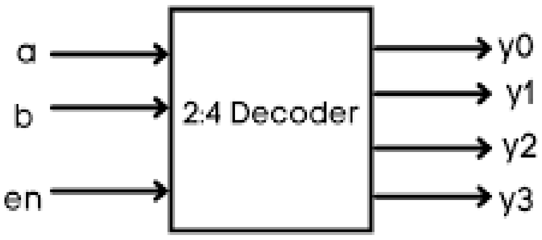


A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Y = A AND B

2:4 Decoder:

A 2:4 decoder is a combinational logic circuit that takes in a 2-bit input and activates one of the 4 output lines based on the input value. It essentially decodes a binary input into one of the 4 possible output combinations.
The 2:4 decoder consists of 2 input lines (A, B) and 4 output lines (Y0, Y1, Y2, Y3).
Each output line corresponds to a specific input combination, representing aunique binary value.
The input lines (A,B) can be set to either 0 or 1, representing the binary values 0 and 1, respectively. The combination of these three input lines determines which output line will be activated or turned on.

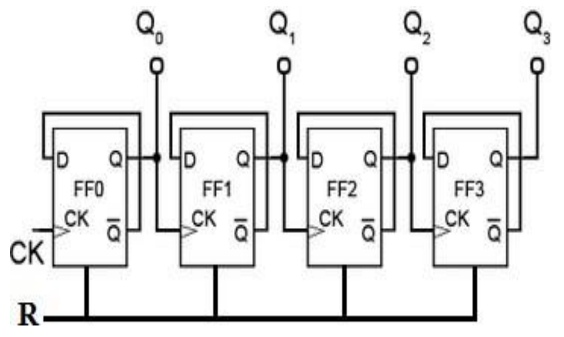


For example, if the input combi- nation is A=0, B=1 the decoder will activate output line Y2. Each input combination corresponds to a specific output line, following a one-hot encoding scheme.
2:4 decoders are commonly used in digital systems to select one of multipleoutputs based on a binary input. They are also used as building blocks for morecomplex circuits, such as multiplexers and address decoders.

4 BIT UP COUNTER:

A four-bit up counter is a digital circuit that counts sequentially from 0 to 15 (in binary) in an upward direction. It con- sists of four flip-flops, each representing a bit position, and combinational logic that generates the input signals to the flip-flops. The counter starts at 0000 (0 in decimal) and increments by 1

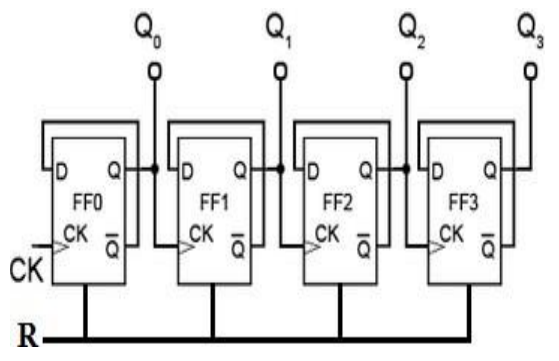
for each clock cycle. As the clock signal pulses, the flip-flops store and propagate the current count value, resulting in a binary sequence from 0000 to 1111. Once the counter reaches its maximum value, it rolls over back to 0000 and continues counting.
The combinational logic used in a four-bit up counter is responsible for generating the input signals to the flip- flops. These signals are derived from the clock signal and the outputs of the flip-flops, ensuring the correct sequencing and incrementation of the counter.
To implement a four-bit up counter, you would typically use D flip-flops as the storage elements for each bit. The clock signal serves as the input to the flip-flops, causing them to latch the current value on the rising edge of the clock. The output of each flip-flop is connected to the next flip-flop's input, creating a ripple effect that propagates the count value.



The 4-bit up counter is a fundamental building block in digital circuit design and finds applications in various fields, including digital clocks, timers, and control systems. It offers a simple and efficient solution for counting and tracking events in a sequential manner.

4-BIT DOWN COUNTER

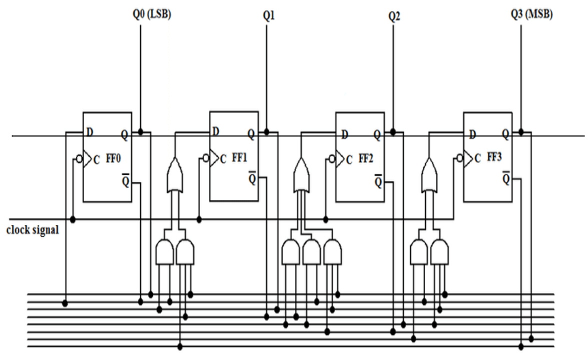
A 4-bit down counter is a digital circuit that counts sequentially from 15 to0 (in binary) in a downward direction. Similar to the 4-bit up counter, it consists of 4 flip-flops and combinational logic that generates the input signals to the flip-flops.
The counter starts at 1111 and decrements by 1 for each clock cycle. As the clock signal pulses, the flip-flops store and propagate the current count value, resulting in a binary sequence from 1111 to 0000. Once the counter reaches its minimum value, it rolls over back to 1111 and continues counting in a downward direction.



The combinational logic used in a 4-bit down counter is responsible for generating the input signals to the flip-flops, similar to the up counter. These signals are derived from the clock signal and the outputs of the flip-flops, ensuring the correct sequencing and decrementation of the counter.
To implement a 4-bit downcounter, you would typically use D flip-flops as the storage elements for each bit, similar to the up counter. The clock signal serves as the input to the flip-flops, causing them to latch the current value on the rising edge of the clock. The output of each flip-flop is connected to the next flip-flop's input, creating a ripple effect that propagates the count value in the downward direc-tion.
The 4-bit down counter is also a fundamental component in digital circuit design and finds applications in various scenarios where counting or decrementing is required, such as countdown timers, data processing, and control systems. It provides a sim-ple and efficient solution for sequential counting in a descending order.

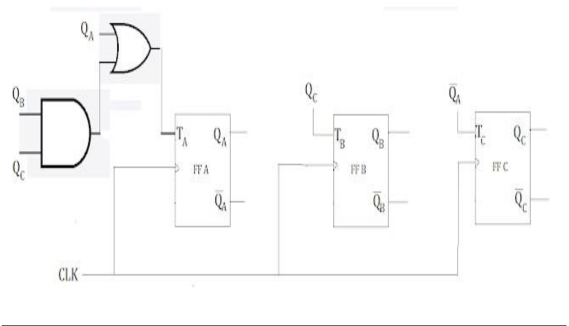
MOD-10 COUNTER

A MOD 10 counter is a digital circuit that counts sequentially from 0 to 9 and then repeats the counting cycle. A MOD 10 counter typically consists of 4 flip-flops and combinational logic that generates the input signals tothe flip-flops. The flip-flops store the current count value, while the combinational logic determines the next count value based on the current count and input signals. At the start, the MOD 10 counter is set to 0000 .On each clock cycle, the counter increments by one. Once the count reaches 9, it wraps back to 0, completing the cycle. This behavior gives the MOD 10 counter its name.
The combinational logic in a MOD 10 counter generates the input signals to the flip-flops. These signals are derived from the current count value and the clock signal. The logic determines the next count value based on the current count and the clock signal, ensuring the correct sequencing and reset behavior.
Implementing a MOD 10 countertypically involves using D flip-flops as the storage elements for each bit. The clock signal serves as the input to the flip-flops, causing them to latch the current count value on the rising edge of the clock. The output of each flip-flop is connected to the next flip-flop's input,creating a ripple effect that propagatesthe count value.
MOD 10 counters are widely used in applications that require a cycle of ten states, such as decimal counters, digital clocks, frequency dividers, and various control systems. They provide a straightforward and efficient solutionfor counting in a modulo 10 sequence, allowing for repeated and controlled operations.



MOD-5 COUNTER

A MOD 5 counter is a digital circuitthat counts sequentially from 0 to 4 and then repeats the countingcycle. It is a type of counter that follows a modulo-5 counting sequence.
A MOD 5 counter typically consists of 3 flip-flops and combinational logic that generates the input signalsto the flip-flops. The flip-flops store the current count value, while the com-binational logic determines the next count value based on the current countand input signals.
At the start, the MOD 5 counter isset to 000. On each clock cycle, the counter increments by one,moving through the sequence 0, 1, 2, 3, 4. Once the count reaches 4, it wraps back to 0, completing the cycle. This behavior gives the MOD 5 counter itsname, as it counts modulo 5.The combinational logic in a MOD5 counter generates the input signals to the flip-flops. These signals are derivedfrom the current count value and the clock signal. The logic determines the next count value based on the current count and the clock signal, ensuring the correct sequencing and reset behavior.
To implement a MOD5 counter,D flip-flops are commonly used



as the storage elements for each bit. The clocksignal serves as the input to the flip- flops, causing them to latch the current count value on the rising edge of the clock. The output of each flip-flop is connected to the next flip-flop’s input, creating a ripple effect that propagates.

WORKING

I have designed an 4-in-one counter using RTL coding in Verilog, utilizing a 2:4 decoder, AND gates, and multiple counters.
Verilog HDL is a hardware description language used for designing digital systems. It allows hierarchical and behavioral modeling, enabling the description of circuit functionality and interconnections. Verilog supports simulation, synthesis, testbenches, and verification techniques, making it essential for digital design.

Here’s a step-by-step explanation ofhow my project works:

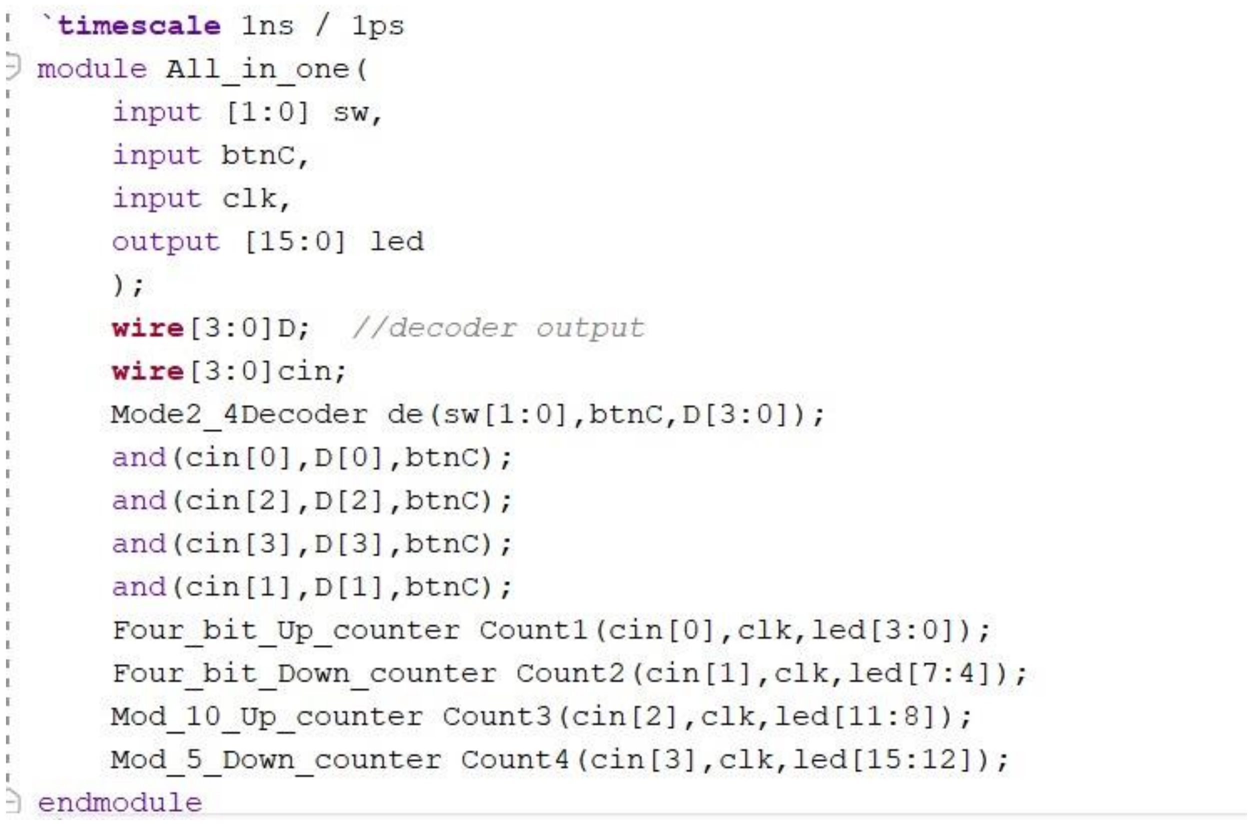
- **Input Signals:** We have 2 input signals (A, B) that act as control signals for the counters.These signals are used to select which counter(s) to operate and reset.
- **2:4 Decoder:** The 2 input signals (A, B) are connected to a 2:4 decoder. The decoder decodes the input signals into 4 output lines, with each line corresponding to a specific counter.
- **AND Gates:** Each output line of the 2:4 decoder is connected to an AND gate, along with a clear input signal.

  Counters: I have multiple counters in our project, and their outputs are connected to the desired output display or further processing logic. Each counter keeps track of a specific value or counts according to the given sequence.
- **Operation:** When you want to increment a particular counter,you set the corresponding input signals (A, B) to the appropriate values that activate the corresponding output line of the 2:4 decoder.
- **Reset Functionality:** To reset a specific counter, you control the clear input signal using the AND gate. The AND gate receives the output from the corresponding output line of the 2:4 decoder and a clear input signal. If the output of the AND gate is low (0),it indicates that the conditions for resetting the counter have been met. In this case, the clear input signal activates, resetting the counter to its initial value.

By selecting the appropriate input combination, I activate the corresponding counter, and its output is displayed on the connected display device. The counters increment or decrement according to their defined behavior, allowing you to visualize and track various counting sequences on the displays.

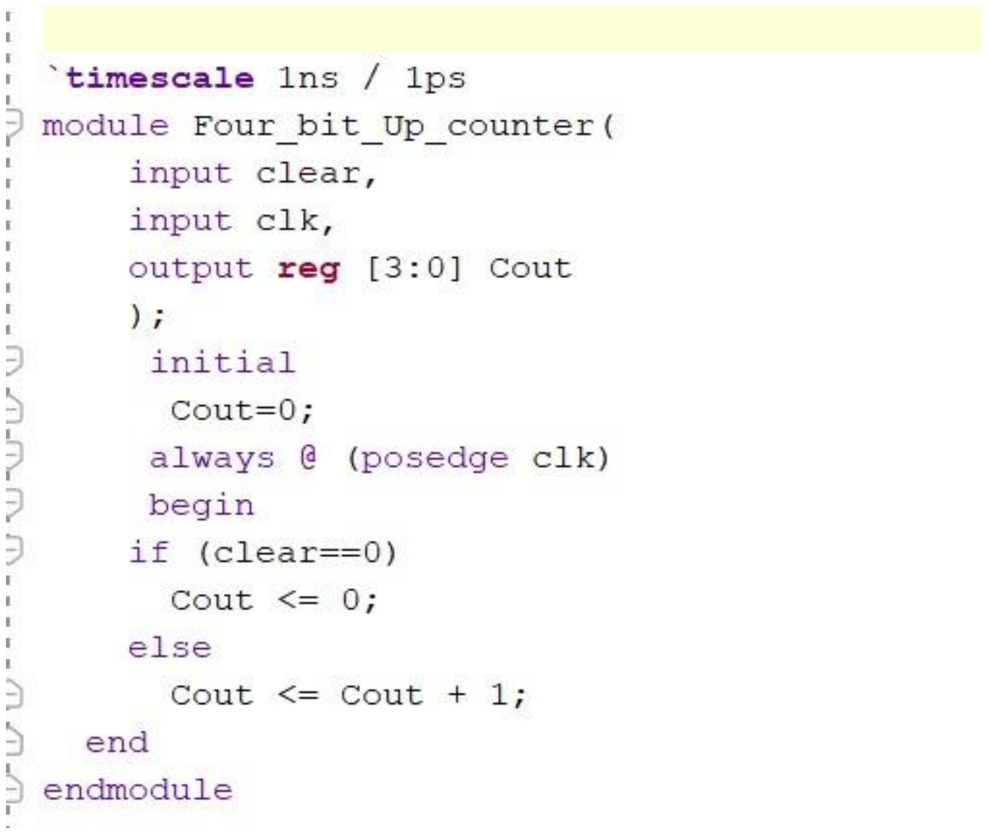
VERILOG CODES

MAIN CODE



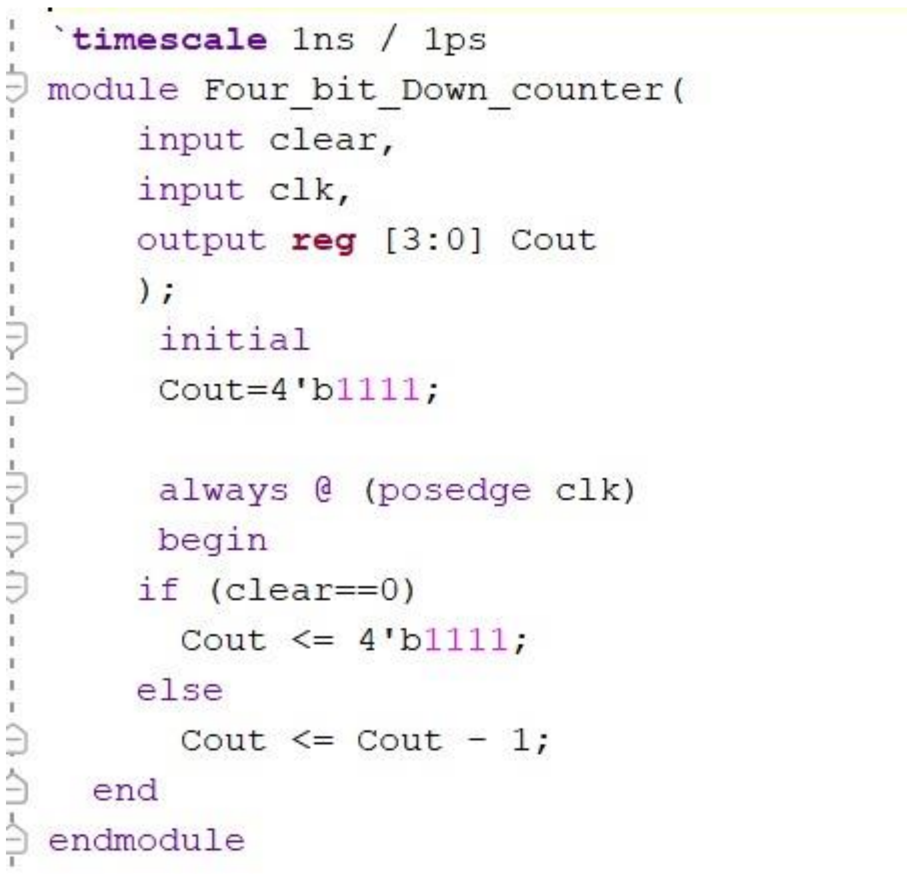
```verilog
`timescale 1ns / 1ps
module All_in_one(
    input [1:0] sw,
    input btnC,
    input clk,
    output [15:0] led
    );
    wire[3:0]D;  //decoder output
    wire[3:0]cin;
    Mode2_4Decoder de(sw[1:0],btnC,D[3:0]);
    and(cin[0],D[0],btnC);
    and(cin[2],D[2],btnC);
    and(cin[3],D[3],btnC);
    and(cin[1],D[1],btnC);
    Four_bit_Up_counter Count1(cin[0],clk,led[3:0]);
    Four_bit_Down_counter Count2(cin[1],clk,led[7:4]);
    Mod_10_Up_counter Count3(cin[2],clk,led[11:8]);
    Mod_5_Down_counter Count4(cin[3],clk,led[15:12]);
endmodule
```

4-BIT UP COUNTER



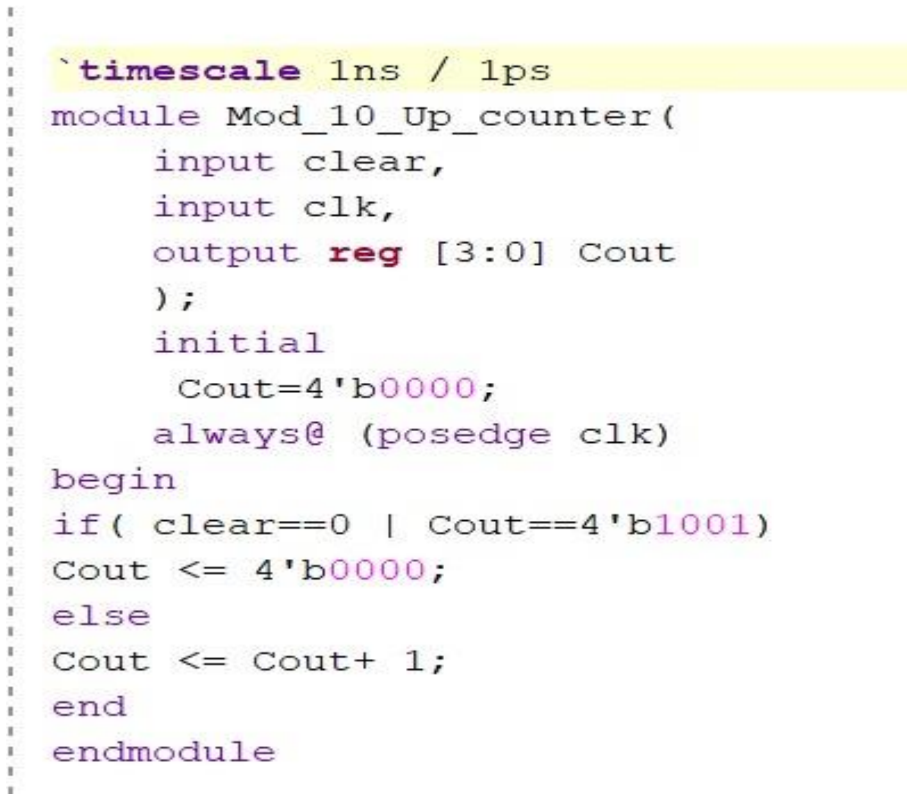
```verilog
`timescale 1ns / 1ps
module Four_bit_Up_counter(
    input clear,
    input clk,
    output reg [3:0] Cout
    );
     initial
      Cout=0;
     always @ (posedge clk)
     begin
    if (clear==0)
      Cout <= 0;
    else
      Cout <= Cout + 1;
  end
endmodule
```

4-BIT DOWN COUNTER



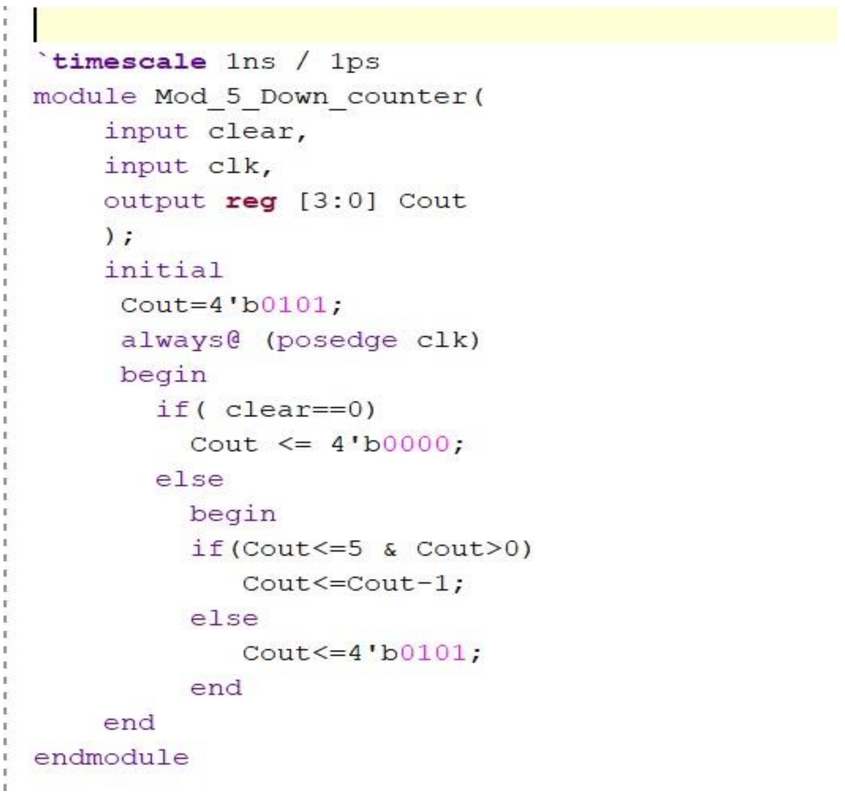
```
`timescale 1ns / 1ps
module Four_bit_Down_counter(
    input clear,
    input clk,
    output reg [3:0] Cout
    );
     initial
     Cout=4'b1111;

     always @ (posedge clk)
     begin
    if (clear==0)
      Cout <= 4'b1111;
    else
      Cout <= Cout - 1;
  end
endmodule
```

MOD-10 UP COUNTER



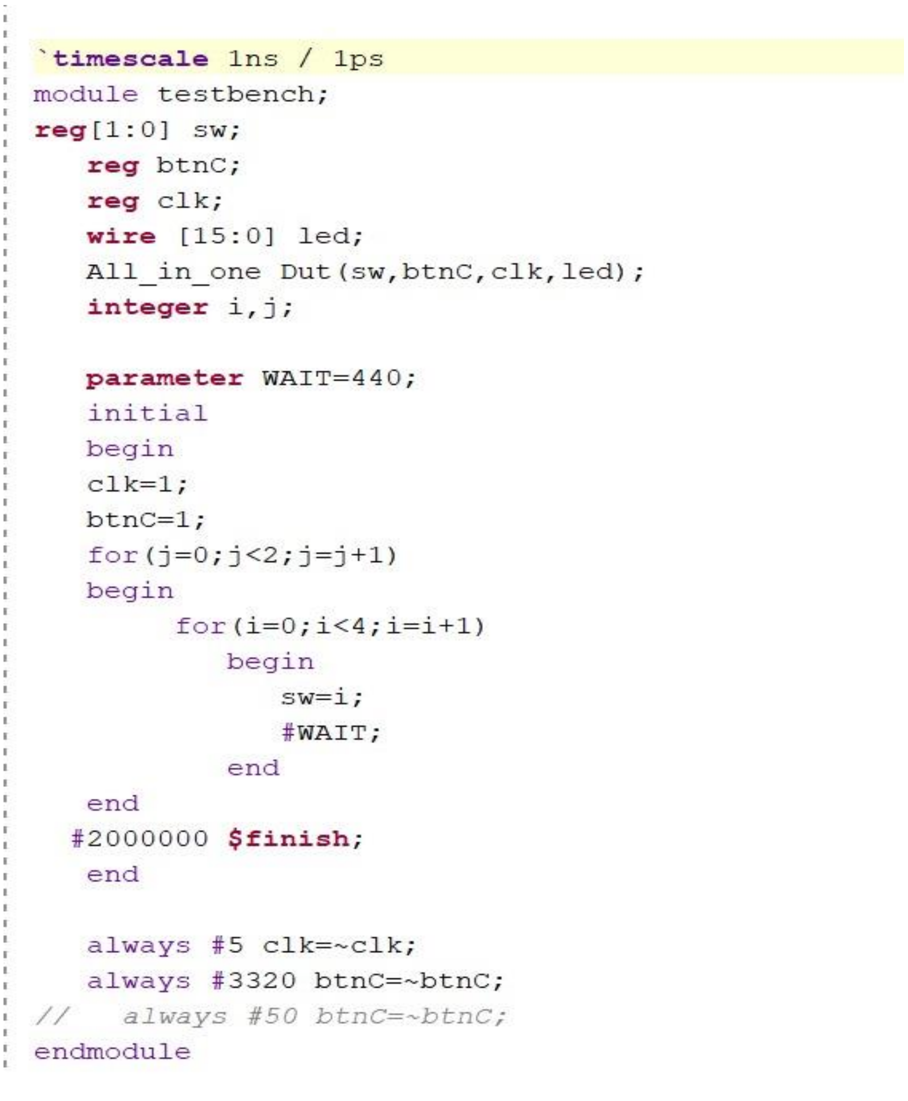
```
`timescale 1ns / 1ps
module Mod_10_Up_counter(
     input clear,
     input clk,
     output reg [3:0] Cout
     );
     initial
      Cout=4'b0000;
     always@ (posedge clk)
begin
if( clear==0 | Cout==4'b1001)
Cout <= 4'b0000;
else
Cout <= Cout+ 1;
end
endmodule
```

MOD-5 DOWN COUNTER



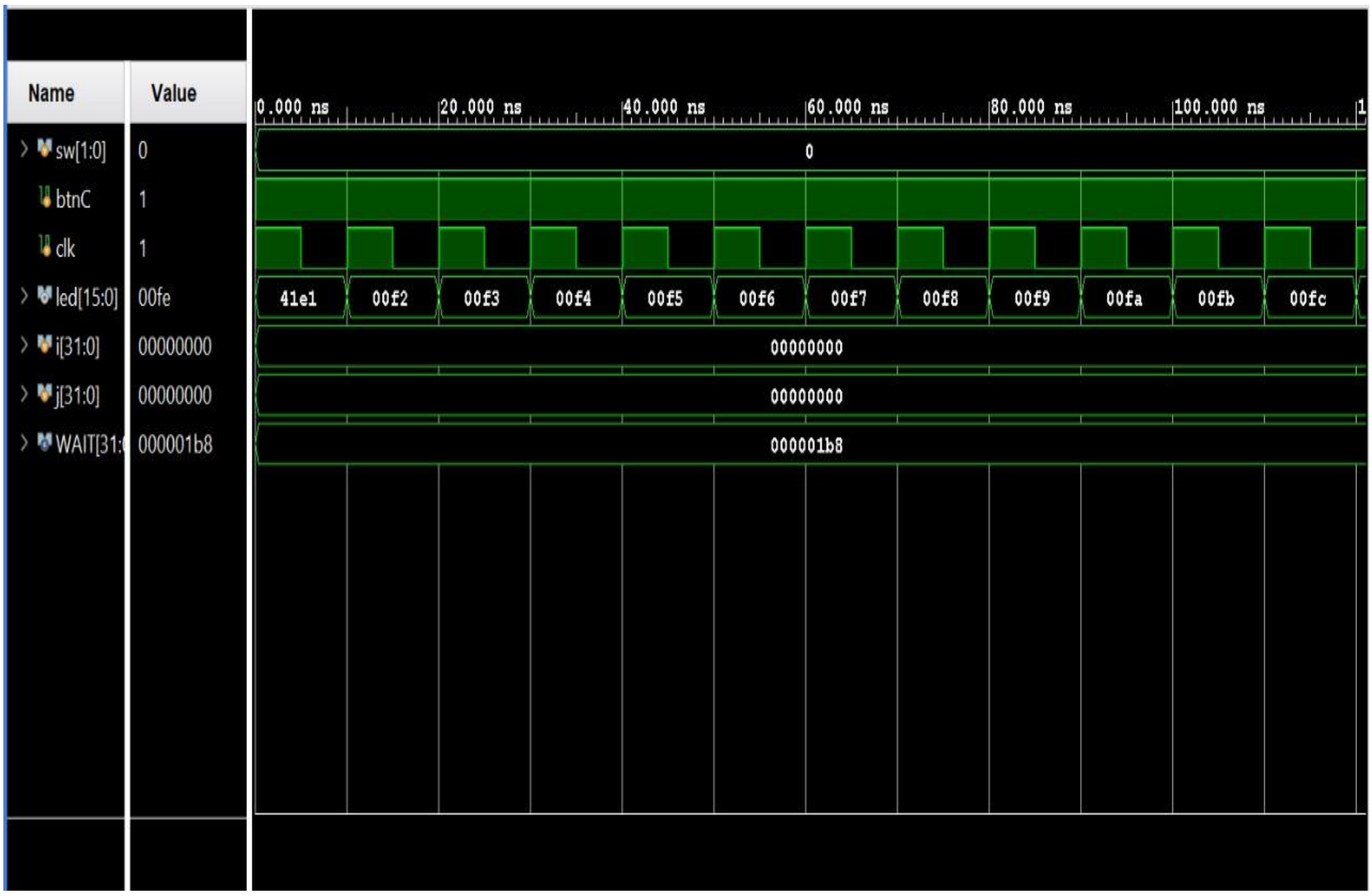
```
`timescale 1ns / 1ps
module Mod_5_Down_counter(
    input clear,
    input clk,
    output reg [3:0] Cout
    );
    initial
     Cout=4'b0101;
     always@ (posedge clk)
     begin
       if( clear==0)
         Cout <= 4'b0000;
       else
         begin
         if(Cout<=5 & Cout>0)
            Cout<=Cout-1;
         else
            Cout<=4'b0101;
         end
    end
endmodule
```

TESTBENCH CODE



```
`timescale 1ns / 1ps
module testbench;
reg[1:0] sw;
   reg btnC;
   reg clk;
   wire [15:0] led;
   All_in_one Dut(sw,btnC,clk,led);
   integer i,j;

   parameter WAIT=440;
   initial
   begin
   clk=1;
   btnC=1;
   for(j=0;j<2;j=j+1)
   begin
        for(i=0;i<4;i=i+1)
           begin
              sw=i;
              #WAIT;
           end
   end
  #2000000 $finish;
   end

   always #5 clk=~clk;
   always #3320 btnC=~btnC;
//   always #50 btnC=~btnC;
endmodule
```

SIMULATION


Name
Value
sw[1:0]
0
btnC
1
clk
1
led[15:0]
00fe
i[31:0]
00000000
j[31:0]
00000000
WAIT[31:
000001b8
0.000 ns
20.000 ns
40.000 ns
60.000 ns
80.000 ns
100.000 ns
0
41e1
00f2
00f3
00f4
00f5
00f6
00f7
00f8
00f9
00fa
00fb
00fc
00000000
00000000
000001b8

CONCLUSION

In conclusion, the project aimed to implement an ”4-in-one” counter using Verilog HDL. The project involved designing and testing various counters, including a 4-bit up counter, 4-bit down counter, mod 10 counter, mod 5 counter.

The implemented design utilized a 2:4 decoder to access the different counters based on the input mode. An AND gate with a clear input was used to reset the counters when the output of the AND gate was 0.

The provided testbench code simulated the behavior of the counters by generating clock signals, setting/resetting the preset signal for the ring counter, and controlling the mode selection. The simulation verified the functionality of the counters and their expected behavior in different modes.

Overall, the project successfully demonstrated the implementation of various counters using structural Verilog coding and validated their operation through simulation.

REFERENCES

- **”Digital Design and Computer Architecture”** by David Harris and Sarah Harris: This book provides a comprehensive introduction to digital design principles and includes a section on Verilog HDL.

- **”Verilog HDL: A Guide to Digital Design and Synthesis”** by Samir Palnitkar:This book offers a detailed explanation of Verilog HDL and covers both the basics and advanced topics of digital design using Verilog.

- **”RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability”** by Pong P. Chu: Although this book focuses on VHDL, it provides valuable insights into hardware design concepts and coding techniques that are applicable to Verilog HDL as well.

- **Xilinx Documentation:** Xilinx, the provider of the FPGA platform you men- tioned, offers extensive documentation and application notes on Verilog HDL andtheir design tools. You can refer to their official website for resources specific to your Xilinx FPGA.

MEMBERS

- **VANASHREE PARATE (BT21ECE084)**
- **ANJALIKA AGARWAL (BT21ECE095)**
- **ANUSHKA CHINTAWAR (BT21ECE096)**
- **CHAITRA GURUVELLI (BT21ECE130)**