ECS 409: Computer Organization – Assignment 4

Adheesh Trivedi

2025-09-18

Problem 1: Moore FSM

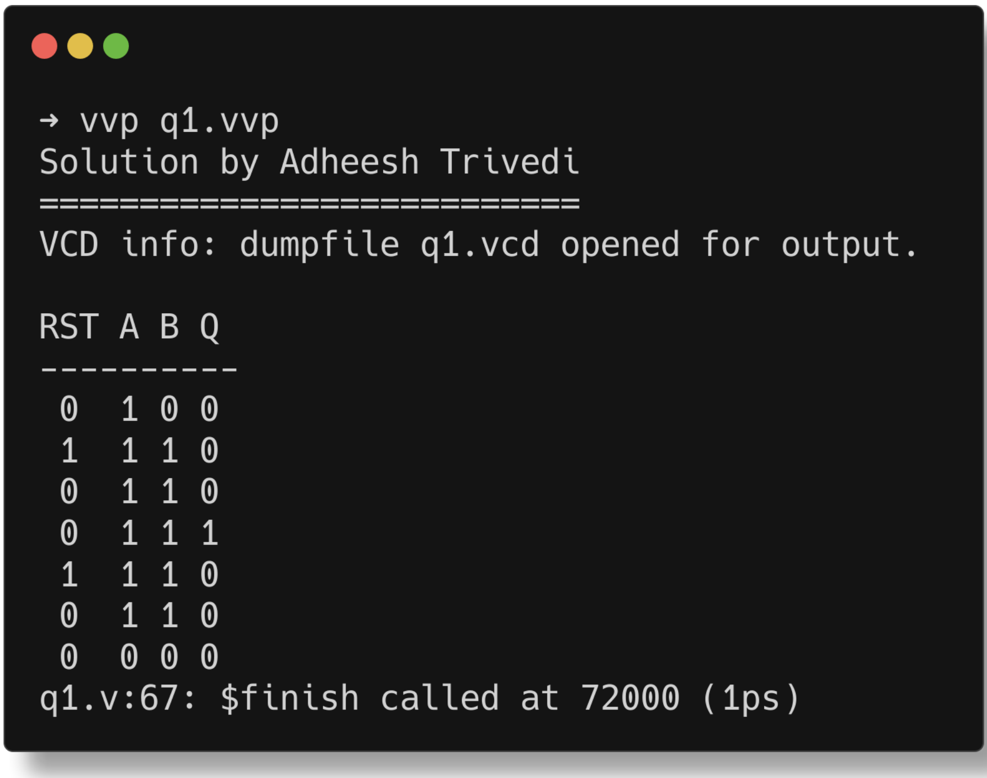
Moore state machine per spec; output depends only on state, reset returns to initial.

```verilog
`timescale 1ns / 1ps

module MOORE_FSM ( input A, B, CLK, RST, output Q );

  parameter S0 = 2'b00;
  parameter S1 = 2'b01;
  parameter S2 = 2'b10;

  reg [1:0] state, nextstate;

  initial state = S0;

  always @(posedge CLK, posedge RST)
    if (RST) state <= S0;
    else
      state <= nextstate;

  always @(*)
    case (state)
      S0: nextstate = A ? S1 : S0;
      S1: nextstate = B ? S2 : S0;
      S2: nextstate = S0;
      default: nextstate = S0;
    endcase

  assign Q = ( state == S2 );

endmodule

module TEST();

  reg A, B, CLK, RST;
  wire Q;

  MOORE_FSM fsm (A, B, CLK, RST, Q);

  initial CLK = 0;
  always begin
    #5; CLK = ~CLK;
  end

  initial begin
    $display("Solution by Adheesh Trivedi");
    $display("===========================");
    $dumpfile("q1.vcd");
    $dumpvars(0, TEST);

    $display("");
    $display("RST A B Q");
    $display("----------");
    $monitor(" %b  %b %b %b", RST, A, B, Q);
    // S0
    RST = 0; A = 1; B = 0; #10; // S1
    RST = 1; B = 1; #10; // Q = 0 if reset
    RST = 0; #10; // S1
    #12; // S2 => Q = 1
    RST = 1; #5; // S0 => Q = 0
    RST = 0; #3;
    A = 0; B = 0; #10; // S1
    #12; // Stays in S1

    $finish;
  end
endmodule
```

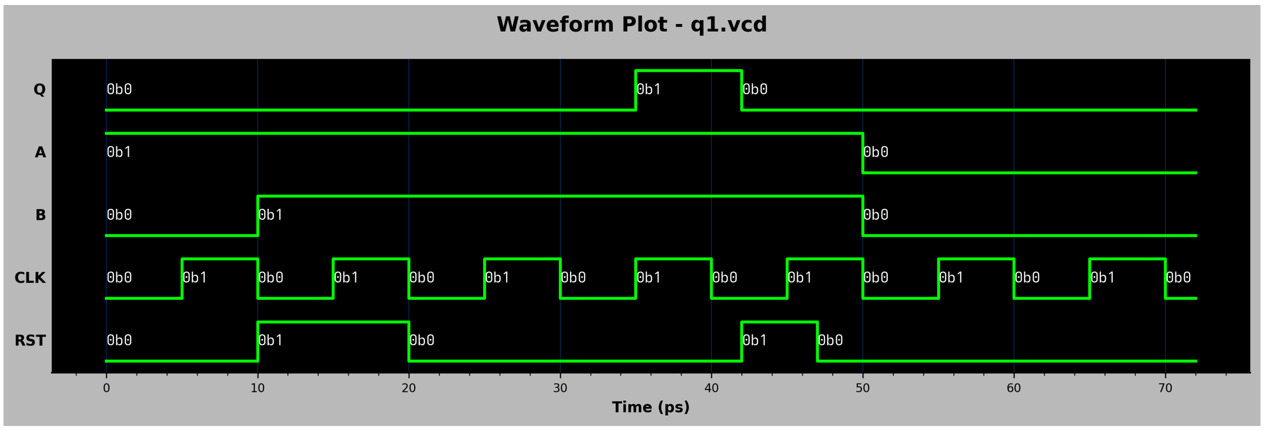
Moore FSM Implementation

1.1 Simulation Output

Output Q asserted only after required state sequence; reset clears immediately. Trace matches transitions in terminal log (see consistent Q after stable inputs).



```
→ vvp q1.vvp
Solution by Adheesh Trivedi
=============================
VCD info: dumpfile q1.vcd opened for output.

RST A B Q
----------
 0  1 0 0
 1  1 1 0
 0  1 1 0
 0  1 1 1
 1  1 1 0
 0  1 1 0
 0  0 0 0
q1.v:67: $finish called at 72000 (1ps)
```



Problem 2: Mealy FSM

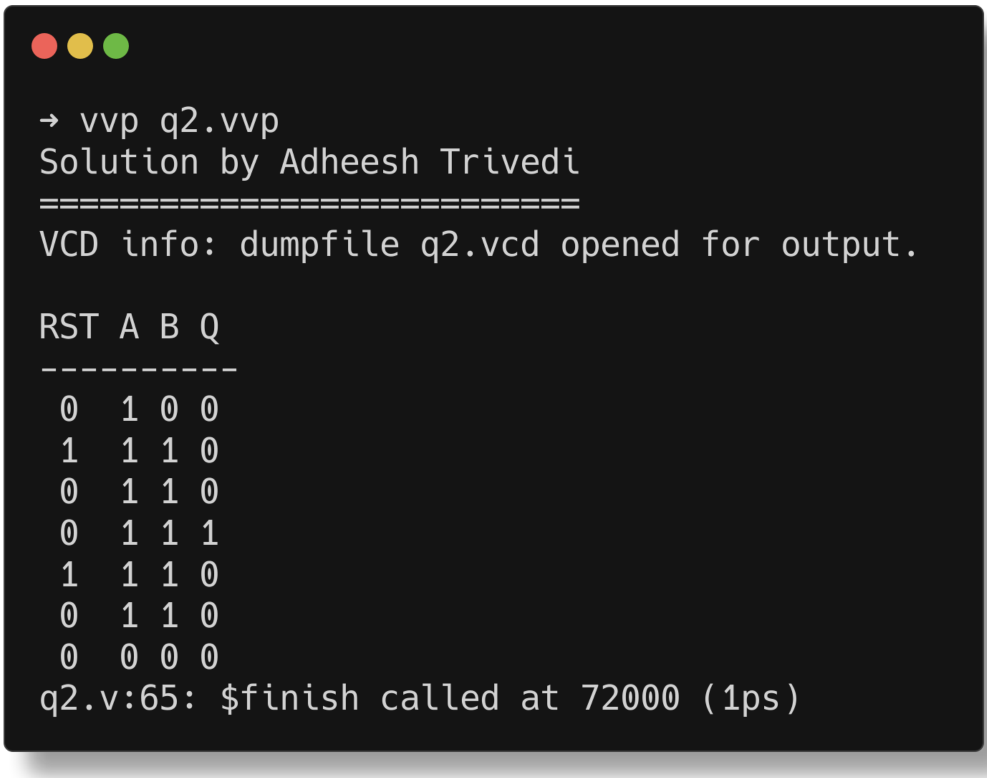
Mealy machine: output reacts instantly to input changes plus current state.

```
`timescale 1ns / 1ps

module MEALY_FSM ( input A, B, CLK, RST, output Q );

  parameter S0 = 2'd1;
  parameter S1 = 2'd2;
  parameter S2 = 2'd3;

  reg [1:0] state, nextstate;

  always @(posedge CLK, posedge RST)
    if (RST) state <= S0;
    else
      state <= nextstate;

  always @(*)
    case (state)
      S0: nextstate = A ? S1 : S0;
      S1: nextstate = B ? S2 : S0;
      S2: nextstate = (A & B) ? S2 : S0;
      default: nextstate = S0;
    endcase

  assign Q = A & B & ( state == S2 );

endmodule

module TEST();

  reg A, B, CLK, RST;
  wire Q;

  MEALY_FSM fsm (A, B, CLK, RST, Q);

  initial CLK = 0;
  always begin
    #5; CLK = ~CLK;
  end

  initial begin
    $display("Solution by Adheesh Trivedi");
    $display("===========================");
    $dumpfile("q2.vcd");
    $dumpvars(0, TEST);

    $display("");
    $display("RST A B Q");
    $display("----------");
    $monitor(" %b  %b %b %b", RST, A, B, Q);

    // S0
    RST = 0; A = 1; B = 0; #10; // S1
    RST = 1; B = 1; #10; // Q = 0 if reset
    RST = 0; #10; // S1
    #12; // S2 => Q = 1
    RST = 1; #5; // S0 => Q = 0
    RST = 0; #3;
    A = 0; B = 0; #10; // S1
    #12; // Stays in S1

    $finish;
  end
endmodule
```

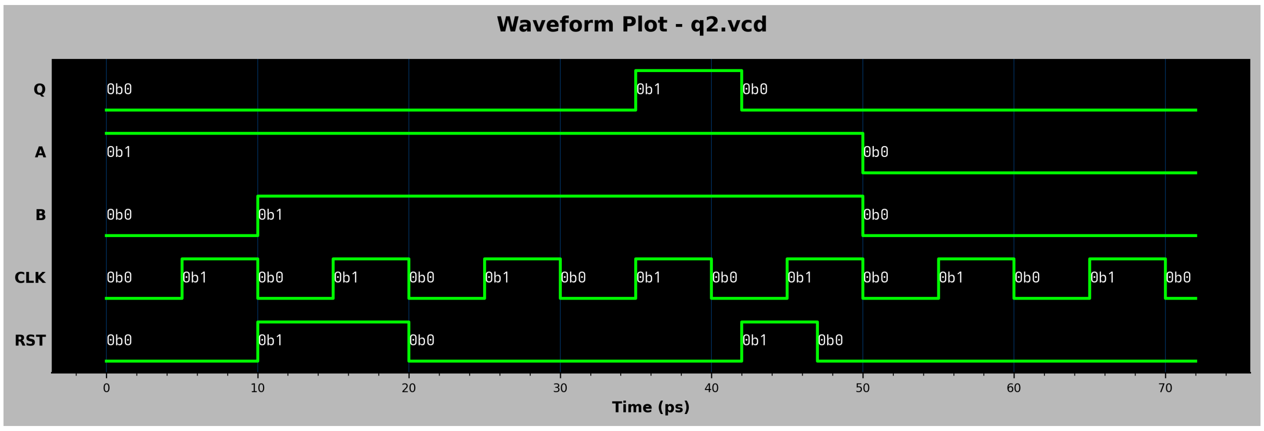
Mealy FSM Implementation

2.1 Simulation Output

Q pulses earlier vs Moore (input-driven), immediate response visible pre-state settle. Reset cycles force Q low; matches log timing stamps and waveform edges.



```
➜ vvp q2.vvp
Solution by Adheesh Trivedi
===========================
VCD info: dumpfile q2.vcd opened for output.

RST A B Q
----------
 0   1 0 0
 1   1 1 0
 0   1 1 0
 0   1 1 1
 1   1 1 0
 0   1 1 0
 0   0 0 0
q2.v:65: $finish called at 72000 (1ps)
```



Problem 3: 1101 Sequence Detector

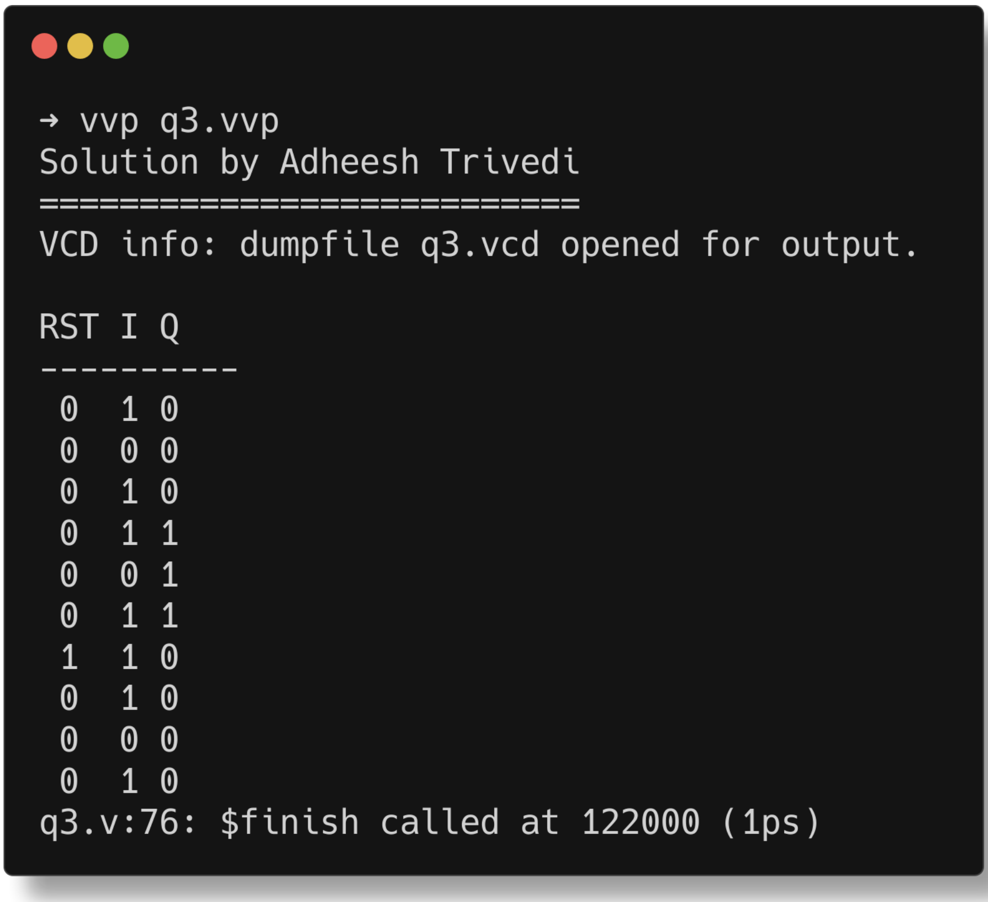
Detects overlapping 1101 patterns; emits 1-cycle Z pulse on detection.

```
`timescale 1ns / 1ps

// Detects the substring "1101" in the input stream
module SUBSTR_FSM ( input I, CLK, RST, output Q );

  parameter S0 = 3'b000;
  parameter S1 = 3'b001;
  parameter S2 = 3'b010;
  parameter S3 = 3'b011;
  parameter S4 = 3'b100;

  reg [2:0] state, nextstate;

  initial state = S0;

  always @(posedge CLK, posedge RST)
    if (RST) state <= S0;
    else
      state <= nextstate;

  always @(*)
    case (state)
      S0: nextstate = I ? S1 : S0;
      S1: nextstate = I ? S2 : S0;
      S2: nextstate = I ? S2 : S3;
      S3: nextstate = I ? S4 : S0;
      S4: nextstate = S4;
      default: nextstate = S0;
    endcase

  assign Q = ( state == S4 );

endmodule

module TEST();

  reg I, CLK, RST;
  wire Q;

  SUBSTR_FSM fsm (I, CLK, RST, Q);

  initial CLK = 0;
  always begin
    #5; CLK = ~CLK;
  end

  initial begin
    $display("Solution by Adheesh Trivedi");
    $display("===========================");
    $dumpfile("q3.vcd");
    $dumpvars(0, TEST);

    $display("");
    $display("RST I Q");
    $display("----------");
    $monitor(" %b  %b %b", RST, I, Q);
    // Input sequence: 1101101
    RST = 0; I = 1; #10; // S1
    I = 1; #10; // S2
    I = 0; #10; // S3
    I = 1; #10; // S4
    I = 1; #10; // S4
    I = 0; #10; // S4
    I = 1; #12; // S4
    RST = 1; #5; // S0
    // Input sequence: 1011
    RST = 0; I = 1; #13; // S1
    I = 0; #10; // S2
    I = 1; #10; // S3
    I = 1; #12; // S4

    $finish;
  end
endmodule
```

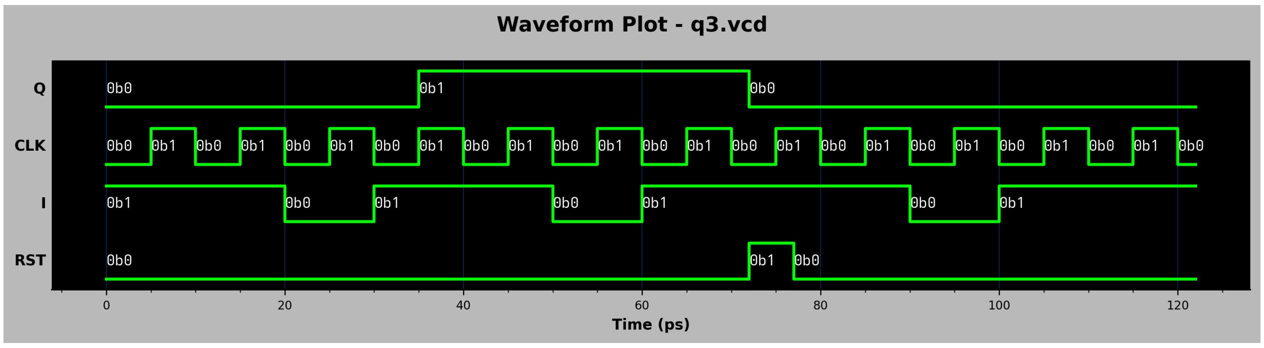
1101 Overlap Detector

3.1 Simulation Output

Z asserted exactly when last bit completes 1101; overlapping occurrences retained. Reset returns to S0; intermediate misses show proper partial-state progression.



```
➜ vvp q3.vvp
Solution by Adheesh Trivedi
===========================
VCD info: dumpfile q3.vcd opened for output.

RST I Q
----------
 0   1 0
 0   0 0
 0   1 0
 0   1 1
 0   0 1
 0   1 1
 1   1 0
 0   1 0
 0   0 0
 0   1 0
q3.v:76: $finish called at 122000 (1ps)
```


Waveform Plot - q3.vcd
Q
CLK
I
RST
0b0
0b1
0b0
0b0 0b1 0b0 0b1 0b0 0b1 0b0 0b1 0b0 0b1 0b0 0b1 0b0 0b1 0b0 0b1 0b0 0b1 0b0 0b1 0b0 0b1 0b0 0b1 0b0
0b1
0b0
0b1
0b0
0b1
0b0
0b1
0b0
0b1
0b0
0
20
40
60
80
100
120
Time (ps)

Problem 4: Modular Equation FSM (2N(PA)+N(PB)≡1 mod4)

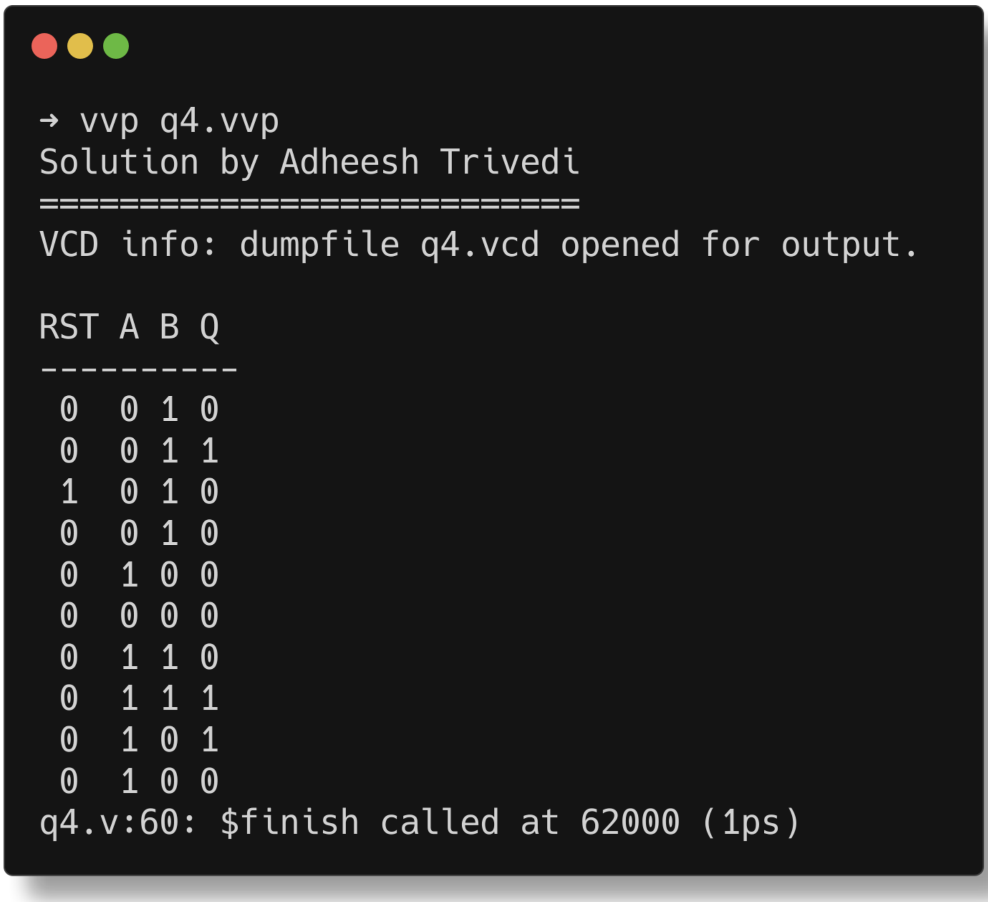
Counts PA (weight2) + PB (weight1) modulo 4, asserts O on residue 1.

```verilog
`timescale 1ns / 1ps

module MODULER_2A_B_FSM ( input A, B, CLK, RST, output O );

  parameter S0 = 2'b00;
  parameter S1 = 2'b01;
  parameter S2 = 2'b10;
  parameter S3 = 2'b11;

  reg [1:0] state, nextstate;

  initial state = S0;

  always @(posedge CLK, posedge RST)
    if (RST) state <= S0;
    else
      state <= nextstate;

  always @(*)
    nextstate = state + 2 * A + B;

  assign O = ( state == S1 );

endmodule

module TEST();

  reg A, B, CLK, RST;
  wire Q;

  MODULER_2A_B_FSM fsm (A, B, CLK, RST, Q);

  initial CLK = 0;
  always begin
    #5; CLK = ~CLK;
  end

  initial begin
    $display("Solution by Adheesh Trivedi");
    $display("===========================");
    $dumpfile("q4.vcd");
    $dumpvars(0, TEST);

    $display("");
    $display("RST A B Q");
    $display("----------");
    $monitor(" %b  %b %b %b", RST, A, B, Q);

    RST = 0; A = 0; B = 1; #12;
    RST = 1; #5;
    RST = 0; #3;
    A = 1; B = 0; #10;
    A = 0; B = 0; #10;
    A = 1; B = 1; #10;
    A = 1; B = 0; #12;

    $finish;
  end
endmodule
```

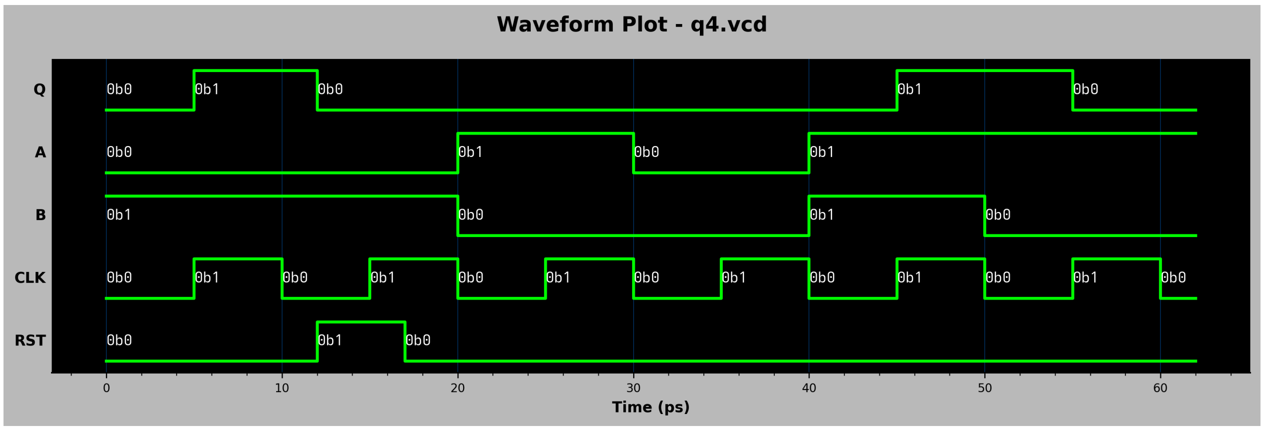
Mod-4 Weighted Counter FSM

4.1 Simulation Output

O high exactly when (2N(PA)+N(PB)) mod4 =1; other cycles low; reset clears counts. State/output pattern in log aligns with theoretical residue progression.



```
➜ vvp q4.vvp
Solution by Adheesh Trivedi
===========================
VCD info: dumpfile q4.vcd opened for output.

RST A B Q
----------
 0   0 1 0
 0   0 1 1
 1   0 1 0
 0   0 1 0
 0   1 0 0
 0   0 0 0
 0   1 1 0
 0   1 1 1
 0   1 0 1
 0   1 0 0
q4.v:60: $finish called at 62000 (1ps)
```


Waveform Plot - q4.vcd
Q
A
B
CLK
RST
0b0
0b1
Time (ps)