Name: Abdul Qadeer
Roll No: EL-02/2025
Advanced Digital System Design Mid 1 Paper

Q1

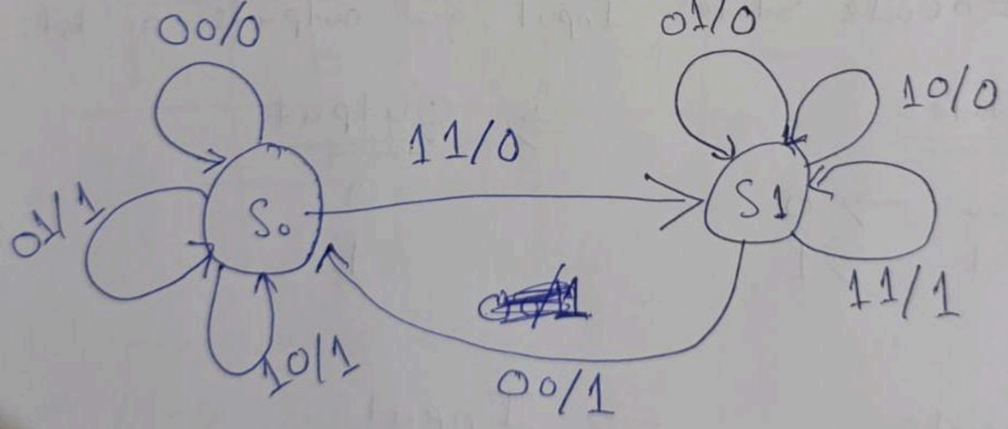
(1) How many states are required?

$\Rightarrow$ we require two states

State 0 (S_0) $\rightarrow$ No Carry

State 1 (S_1) $\rightarrow$ Carry exists.

(2) State Diagram



(6   1 ~ .

(3) Write output and next-state tables

Present State	Input A	Input B	Next State	Output
S0	0	0	S0	0
S0	0	1	S0	1
S0	1	0	S0	1
S0	1	1	S1	0
S1	0	0	S0	1
S1	0	1	S1	0
S1	1	0	S1	0
S1	1	1	S1	1

(4) Encode States, Input, and outputs as bits

⇒ States

S0 → 0
S1 → 1

⇒ Output

0
1

~~⇒ Inputs~~

⇒ Inputs

A	B
0	0
0	1
1	0
1	1

(5) Determine logic equations for next state and outputs

logic Equations

Cs	A	B	Ns	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Cs → Current State

A, B → Inputs

Ns → Next State

S → output

Next State logic Equation

Sum of product (SOP) equation will be

$$N_s = (B.C_s) + AB + C_s.A$$

$$\boxed{N_s = (A+B)C_s + AB}$$

logic Equation for output

Sum of product (SOP) equation for output is :

~~$S = \overline{C_s}\,\overline{A}B +$~~

$$S = \overline{C_s}\,\overline{A}B + \overline{C_s}A\overline{B} + C_s\overline{A}\,\overline{B} + C_sAB$$

$$S = \overline{C_s}(\overline{A}B + A\overline{B}) + C_s(\overline{A}\,\overline{B} + AB)$$

$$\because \overline{A}B + A\overline{B} = A \oplus B$$
$$\because \overline{A}\,\overline{B} + AB = \overline{A \oplus B}$$

$$S = \overline{C_s}\; A \oplus B + C_s\;\overline{A \oplus B}$$

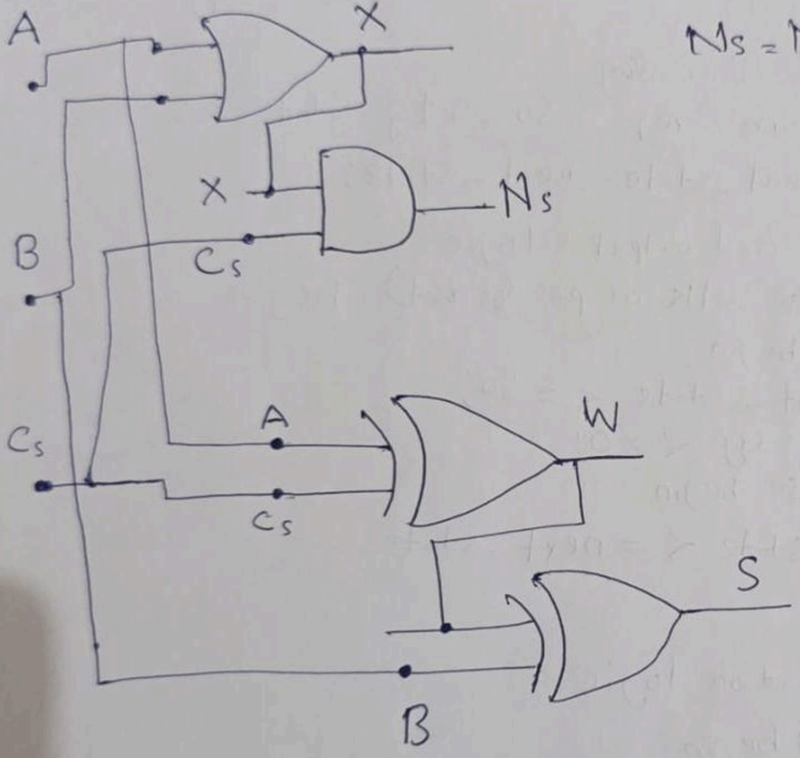
$$\boxed{S = C_s \oplus A \oplus B}$$

(6) Draw the Circuit

Cs = current State, $Ns = (A+B)C_s + AB$

$S = C_s \oplus A \oplus B$

Ns = Next State.



(7) Write RTL Code in verilog or System_verilog

```
module AQ_adder (
        clk, rst, A, B;   //declare inputs
        Output reg sum;   // declare output
        reg Carry;        // declare carry
// Define State Encoding
   typedef enum reg {So, S1} State_t;
   State_t current_state, next_state;
// Next State and output logic
 alway @ (posedge clk or posedge rst) begin
        if (rst) begin
          Current_ state <= So;
              carry <= 0;
        end else begin
        Current_State <= next_State
     end
  end
// State transition logic
  alway @ (*) begin
      case ( current_State)
       So: begin
       sum = A^B;
       Carry = A&B;
       next_State = (A&B) ? S1:So;
  end
      S1:begin
       Sum = A^B^ Carry;
       Carry = (A&B) | ( A & Carry) | (B & Carry);
       next_State = (carry) ? S 1: So;
    end
     endcase
          end
              endmodule
```

EDA Playground Ground Simulation

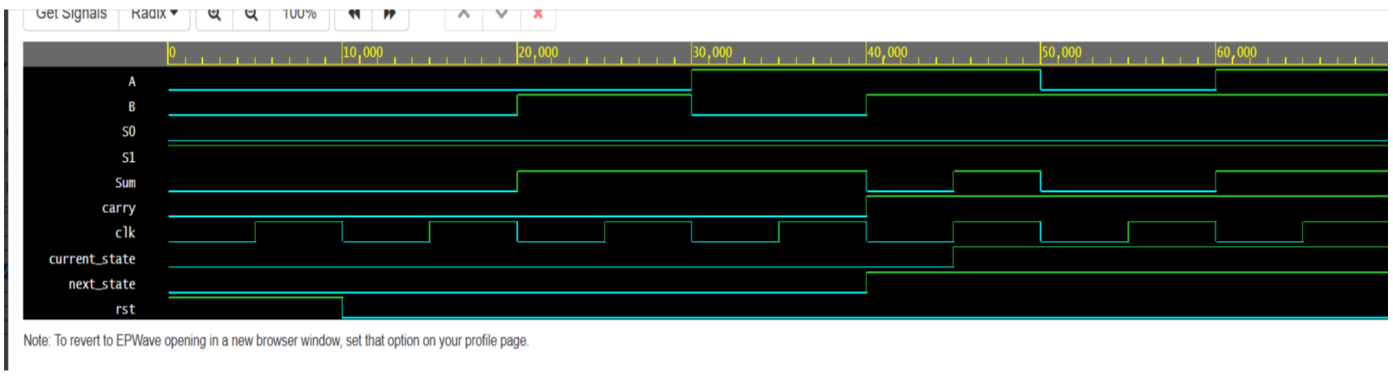
Design Code

```
timescale 1ns / 1ps
module binary_stream_adder (
   input clk, rst, A, B,   // Declare inputs
   output reg Sum          // Declare output as reg
);
   reg carry;              // Declare carry as reg

   // State Encoding using parameters
   parameter S0 = 1'b0, S1 = 1'b1;
   reg current_state, next_state;

   // Sequential logic: state transitions
   always @(posedge clk or posedge rst) begin
      if (rst) begin
         current_state <= S0;
         carry <= 0;
      end else begin
         current_state <= next_state;
      end
   end

   // Combinational logic: next state and output logic
   always @(*) begin
      case (current_state)
         S0: begin
            Sum = A ^ B;      // Sum output = A XOR B
            carry = A & B;    // Carry = A AND B
            next_state = (A & B) ? S1 : S0;
         end
         S1: begin
            Sum = A ^ B ^ carry; // Sum with carry
            carry = (A & B) | (A & carry) | (B & carry); // Updated Carry
            next_state = (carry) ? S1 : S0;
         end
      endcase
   end
endmodule
```

Test Bench Code

```
timescale 1ns / 1ps
module binary_stream_adder_tb;
    reg clk, rst, A, B;
    wire Sum;

    // Instantiate the binary_stream_adder module
    binary_stream_adder uut (
        .clk(clk),
        .rst(rst),
        .A(A),
        .B(B),
        .Sum(Sum)
    );

    // Clock generation
    always #5 clk = ~clk;

    // Test sequence
    initial begin
        $dumpfile("binary_stream_adder_tb.vcd");  // ✅ Generates waveform for
GTKWave
        $dumpvars(0, binary_stream_adder_tb);

        clk = 0; rst = 1; A = 0; B = 0; #10;  // Reset
        rst = 0;

        A = 0; B = 0; #10;
        A = 0; B = 1; #10;
        A = 1; B = 0; #10;
        A = 1; B = 1; #10;

        A = 0; B = 1; #10;
        A = 1; B = 1; #10;

        $display("Simulation Complete");  // ✅ Optional confirmation message
        $finish;
    end
endmodule
```

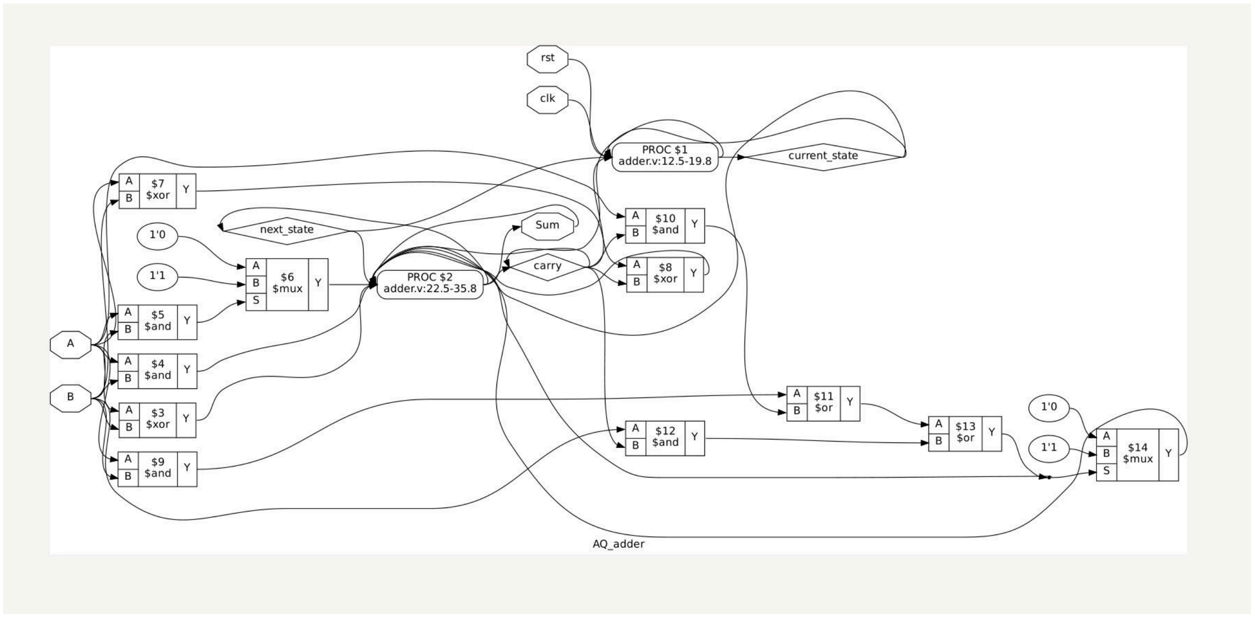
Output Wave



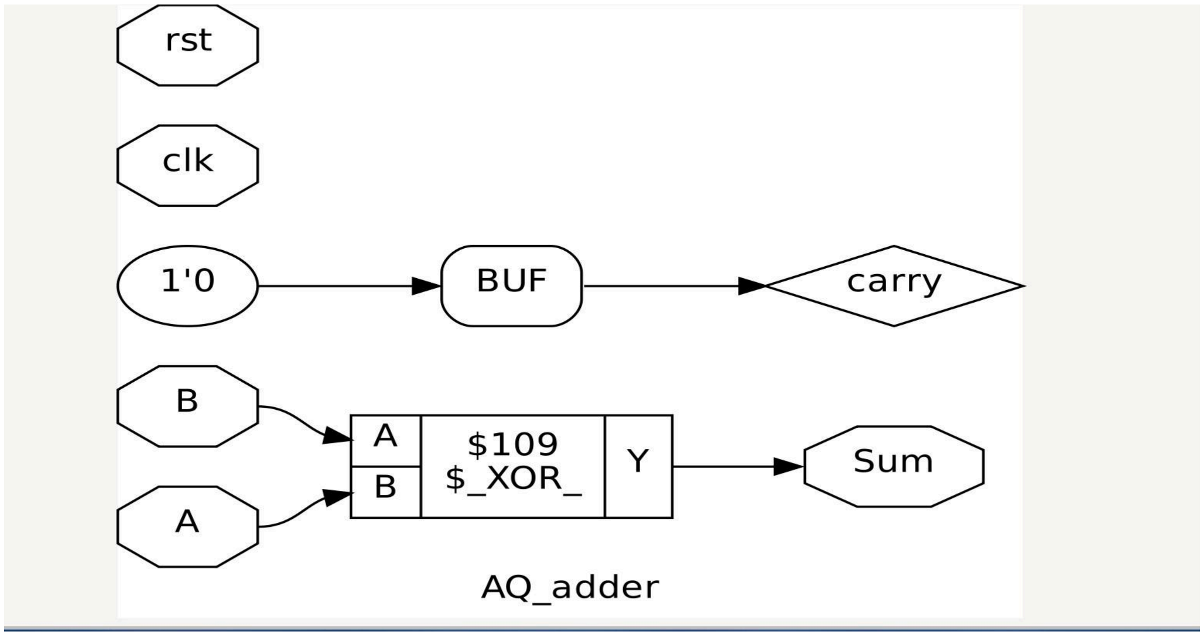
Note: To revert to EPWave opening in a new browser window, set that option on your profile page.

Yosys Synthesis Output

Before Optimization



After Optimization



rst
clk
1'0
BUF
carry
B
A
A
B
$109
$_XOR_
Y
Sum
AQ_adder