**Roll No: 2003063**

**Lab Performance Evaluation [No - 01]**

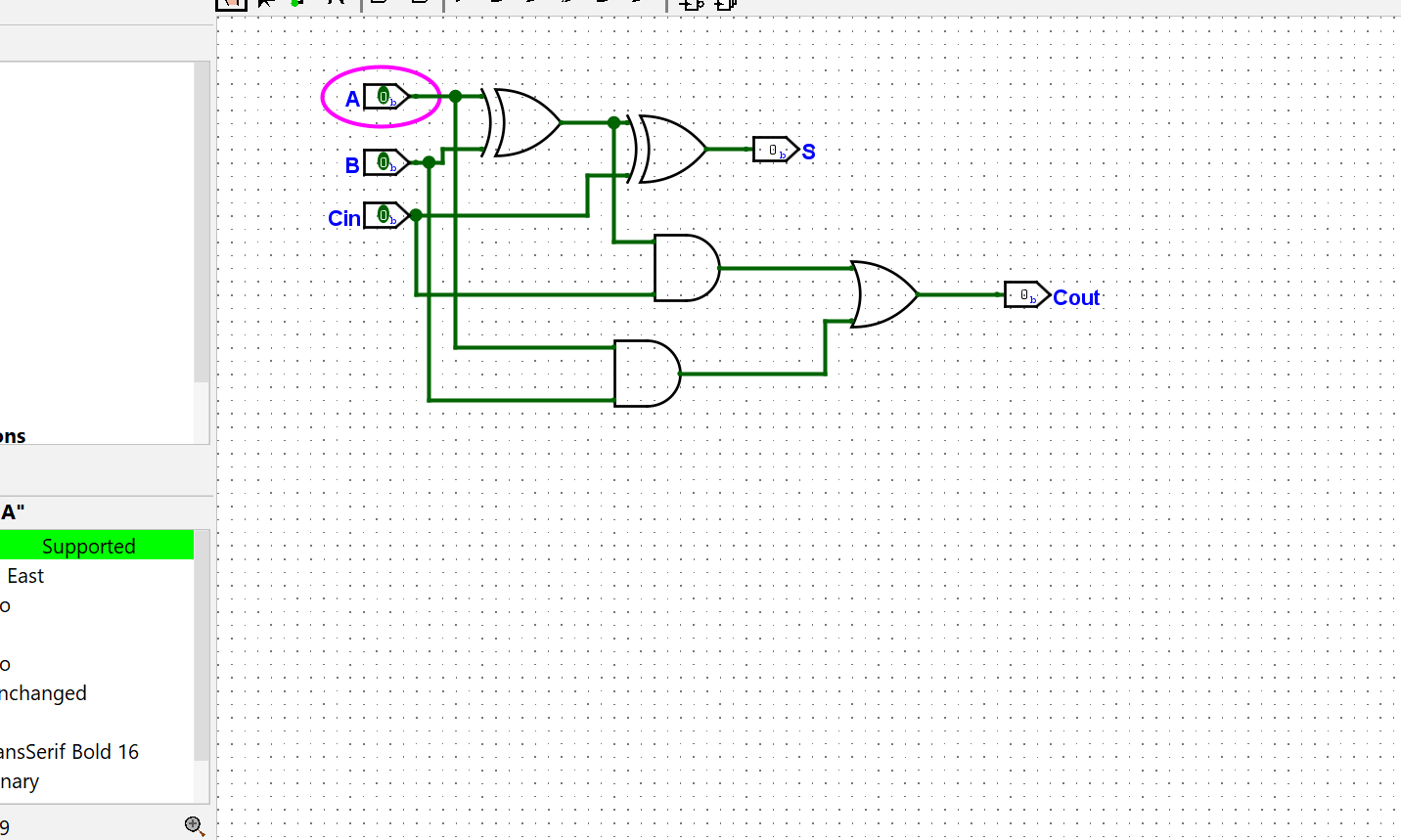
**Lab Task Q[1]**

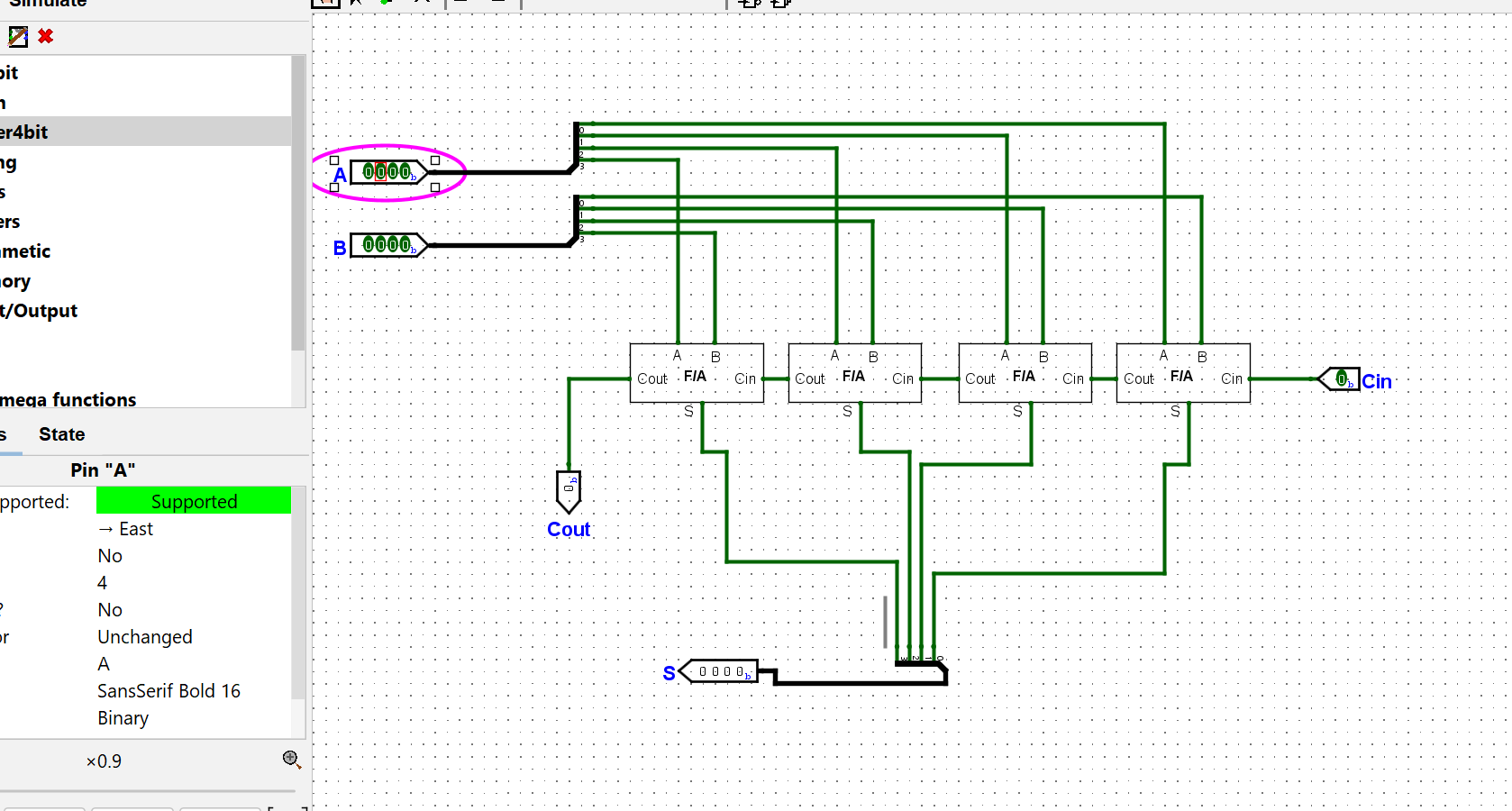
**Question:**

Design a Logic circuit with its Top Level Circuit block (with inputs and outputs pins) with following logic equations: **A = NOT(X.Y).Z**where **NOT(X.Y)** operation will be done using NMOS transistors.

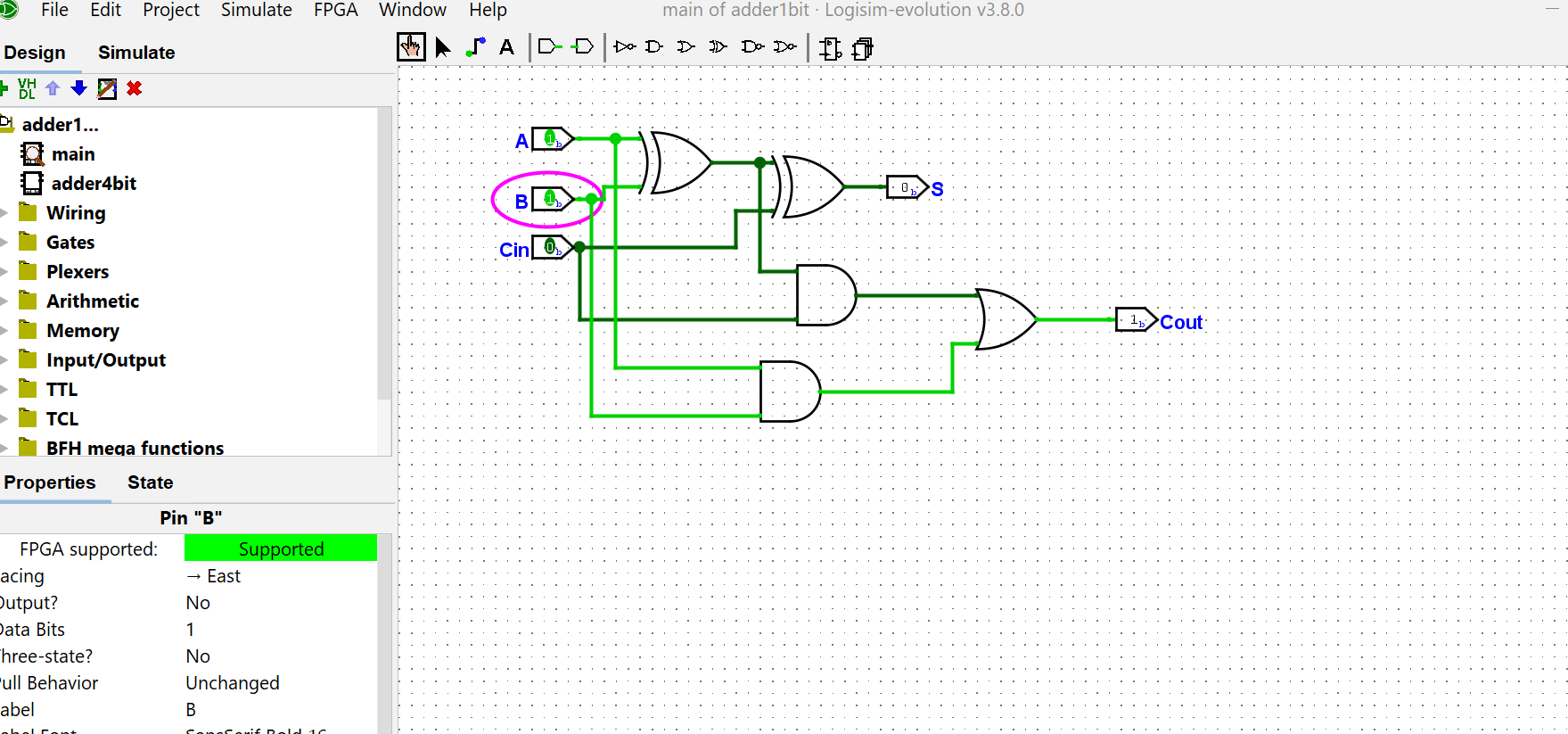
**Notes: I have done adder4bit by using adder1bit**

**Solution (Code/Add all circuits diagrams):**

****

****

**Simulation:**

****

|  |
| --- |
|  |

**Lab Task Q[2]**

**Question:**

Construct an equivalent ARM Assembly Program of following C Program:

**int x = 10; //Do not use Register as variable x**

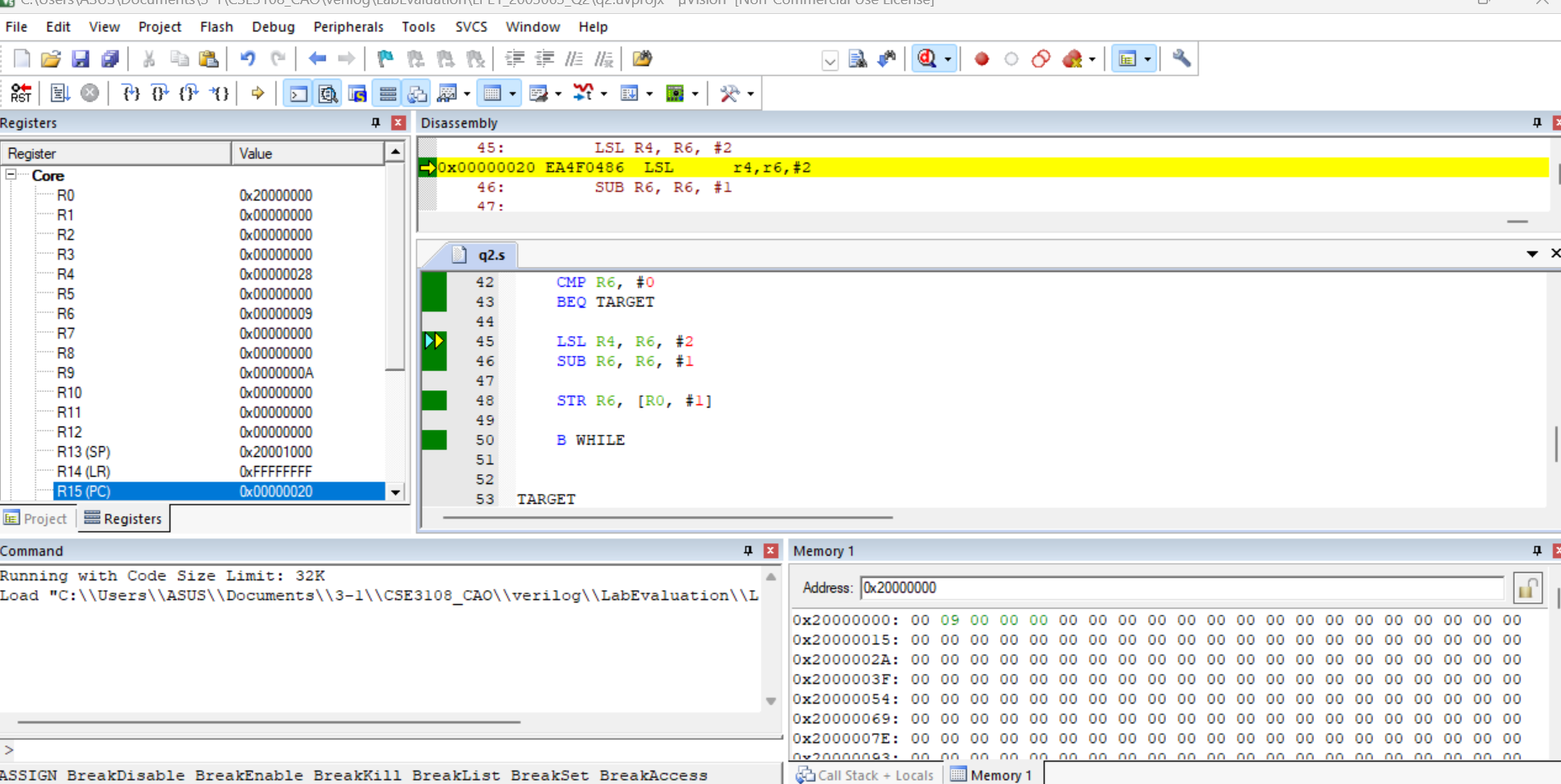
**int y = 0; //Use Register as variable y**

**while ( x > 0 ) {  y=x<<2; x = x-1; }**

**Solution (Code/Add all circuits diagrams):**

|  |
| --- |
| ;;; Directives  PRESERVE8  THUMB    ; Vector Table Mapped to Address 0 at Reset  ; Linker requires \_\_Vectors to be exported  AREA RESET, DATA, READONLY  EXPORT \_\_Vectors    \_\_Vectors  DCD 0x20001000  ; stack pointer value when stack is empty  DCD Reset\_Handler ; reset vector  ALIGN    ; The program  ; Linker requires Reset\_Handler  AREA MYCODE, CODE, READONLY    ENTRY  EXPORT Reset\_Handler      Reset\_Handler  ;;;;;;;;;;User Code Starts from the next line;;;;;;;;;;;;  ; X = 10  MOV R0, #0x20000000  MOV R9, #10    STR R9, [R0, #1]    MOV R4, #0 ; Y = 0      WHILE  LDR R6, [R0, #1]    CMP R6, #0  BEQ TARGET    LSL R4, R6, #2  SUB R6, R6, #1    STR R6, [R0, #1]    B WHILE    TARGET  STOP  B STOP    END ;End of the program |

**Output/Simulation Result (Screenshot/SnapShot):**



**Lab Task Q[3]**

**Question:**

Construct logic circuit will perform following operations on 5 bit inputs a and b to give 5 bit output y:

**case(sel = 11): y = a+b;**

**case(sel = 10): y = a<<b;**

**case(sel = 01): y = a>>b;**

**default: y = b;**

using **if else**statement and synchronous design methodology along with a test bench.

**Solution (Code/Add all circuits diagrams):**

|  |
| --- |
| module sample (      input [1:0] S,      input [4:0] A,      input [4:0] B,      output reg [4:0] Y  );  always @ (\*)      begin          if (S == 2'b11)              Y <= A | B;          else if (S == 2'b10)              Y <= A << B;          else if (S == 2'b01)              Y <= A >> B;          else              Y <= B;        end  endmodule |
| `timescale 1ns/1ps  module sample\_tb ();  reg [1:0] S;  reg [4:0] A;  reg [4:0] B;  wire [4:0] Y;  sample uut (      .S (S),      .A (A),      .B (B),      .Y (Y)  );  initial begin      A = 5'b00101;      B = 5'b01010;      S = 2'b00;      #20;      S = 2'b01;      #20;      S = 2'b10;      #20;      S = 2'b11;  end  initial begin      $monitor ("S = %b, A = %b, B = %b || Y = %b", S, A, B, Y);  end  endmodule |

**Output/Simulation Result (Screenshot/SnapShot):**

