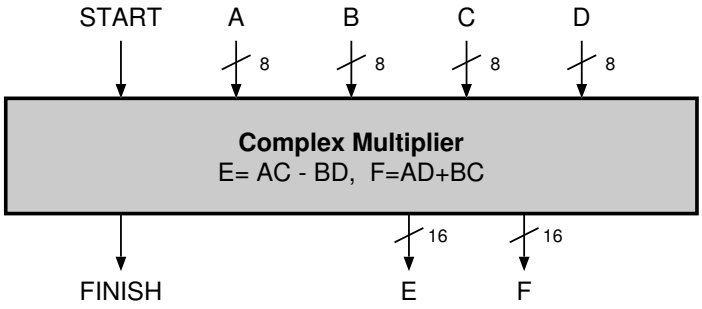
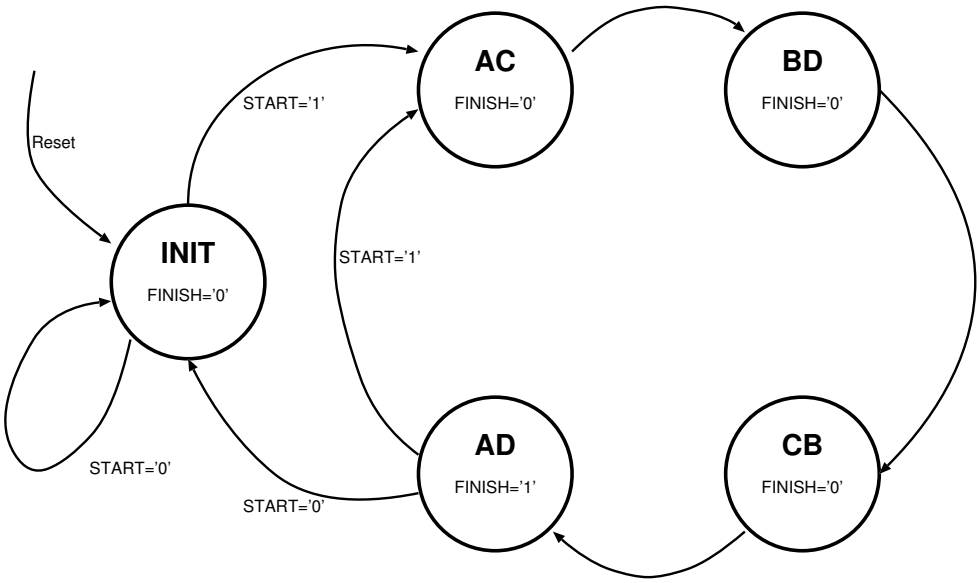
3. In this question we will investigate a circuit that can calculate the multiplication of two complex numbers expressed in the form $(a + bi)$ where a and b are integers expressed in 8-bit two's complement form and i is the complex number which satisfies the equation $i^2 = -1$. As you may remember the multiplication of two complex numbers is

$$\begin{aligned}(e + fi) &= (a + bi) \times (c + di) \\ &= (ac - bd) + (ad + bc)i\end{aligned}$$

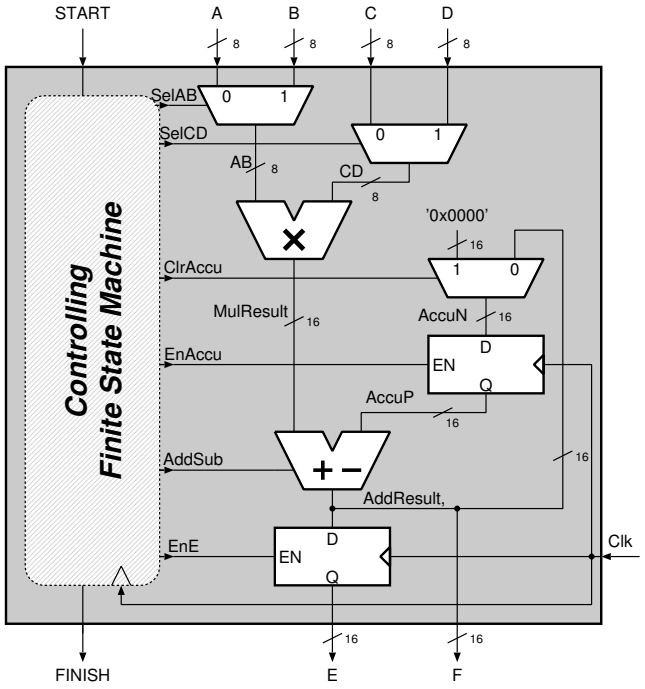
The circuit that we will build has four separate 8-bit inputs (A, B, C, D) for the two complex numbers. A START signal is used to tell the circuit that a new pair of complex numbers are ready to be processed. The circuit has two 16-bit outputs (E, F) for the complex result, and a FINISH signal that tells that the circuit has result has been calculated and the result is ready at outputs E and F.



A colleague of yours has designed an architecture that performs the complex multiplication operation serially using only one multiplier and an adder subtractor. The circuit is controlled by a finite state machine. The following is the state diagram of the state machine. The state names correspond to the operators used.



The circuit block diagram is also given. It can be seen that the FSM controls a series of internal signals. We are interested in determining the correct value of these signals for each state.



(a) (3 points) In your own words describe how the circuit works. For each of the calculation states (AC, BD, CB, AD), explain which operations take place:

When the `Reset` signal is active the circuit moves to the `INIT` state. The `ClrAccu` signal is '0' clearing the accumulator while in this state. The circuit stays in this state as long as `START` signal remains '0'...

Solution:

As soons as `START` is active, we first move to state `AC`. In this state, input `A` and `C` are selected and multiplied. The result goes to the accumulator. In the next state `BD`, this time inputs `B` and `D` are selected. This is subtracted from the accumulator, and the result is written to output register`E`. At the same time the accumulator is cleared. In the cycle `CB`, this time `C` and `B` are selected, multiplied and added to the accumulator. In the last cycle `AD`, inputs `A` and `D` are selected, multiplied and added to the accumulator. The output `FINISH` is set to 1. If the signal `START` is 1, the accumulator is cleared and the system moves to `AC`. Otherwise, the system moves to `INIT` state.

(b) (7 points) Complete the following table, so that we can determine how to design the FSM.

Signal	**Value**	**Description**
SelAB	1	Select input B
SelCD	1	Select input D
ClrAccu	1	Assign the value '0' to the accumulator register input
EnAccu	1	Enable the accumulator register
AddSub	1	Perform Addition
EnE	1	Enable the register for output E

State	START	**NextState**	SelAB	SelCD	ClrAccu	EnAccu	AddSub	EnE	FINISH
INIT	0	INIT	X	X	1	1	X	0	0
INIT	1	AC	X	X	1	1	X	0	0
AC	X	BD	0	0	0	1	1	0	0
BD	X	CB	1	1	1	1	0	1	0
CB	X	AD	1	0	0	1	1	0	0
AD	0	INIT	0	1	1	1	1	0	1
AD	1	AC	0	1	1	1	1	0	1

page intentionally blank

(c) (10 points) The following is the Verilog code that should realize the complex multiplier. In this circuit the FSM is instantiated. The code is divided into 4 parts. Each part contains at most 2 errors. For each part indicate how many errors there are, and re-write the statements with errors so that they are correct.
(*Hint: there are not more than 4 errors in total*)

```
1  module complex (A,B,C,D,E,F, START, FINISH, Clk);
2   input        [7:0] A,B,C,D;
3   output      [15:0] F;
4   input               START, Clk;
5   output              FINISH;
6
7   wire          SelAB, SelCD, EnE;
8   wire          ClrAccu, EnAccu, AddSub;
9   wire [7:0]    AB, CD;
10  wire [15:0]   AccuN;
11  reg  [15:0]   MulResult, AddResult, AccuP;
```

Solution: There is one problem in this section.
The signal E is not defined. It should be defined as `output reg [15:0] E;` since its value is assigned in a process.

```
12  // instantiate FSM, no errors in this statement
13  FSM myFSM (.Clk(Clk), .START(START), .FINISH(FINISH),
14             .SelAB(SelAB), .SelCD(SelCD), .EnE(EnE),
15             .EnAccu(EnAccu), .ClrAccu(ClrAccu),
16             .AddSub(AddSub));
17
18  assign AB    = SelAB ? B : A; // 1:B
19  assign CD    = SelCD ? D : C; // 1:D
20  assign AccuN = ClrAccu ? 16'b0 :AddResult; // 1:Clr
```

Solution: There are no mistakes in this section.

```
21  always @ (AB,CD)
22     MulResult <= AB * CD;
23
24  always @ (MulResult, AccuP)
25     if (AddSub)  AddResult <= MulResult+AccuP; //1:Add
26     else         AddResult <= MulResult-AccuP; //0:Sub
```

Solution: There are two problems in this section:
In line 24, the sensitivity list for the second process does not have the signal `AddSub`
In line 26, the subtraction should be the other way around:
`AccuP - MulResult`.

```
27  always @ (posedge Clk)
28    if (EnE) E <= AddResult;
29
30  always @ (posedge Clk)
31    if (EnAccu) AccuN <= AccuP;
32
33  assign F = AddResult;
34
35 endmodule
```

Solution: There is one problem in this section:
In line 31, the Register definition should be `AccuP<=AccuN`