HomeWork #1

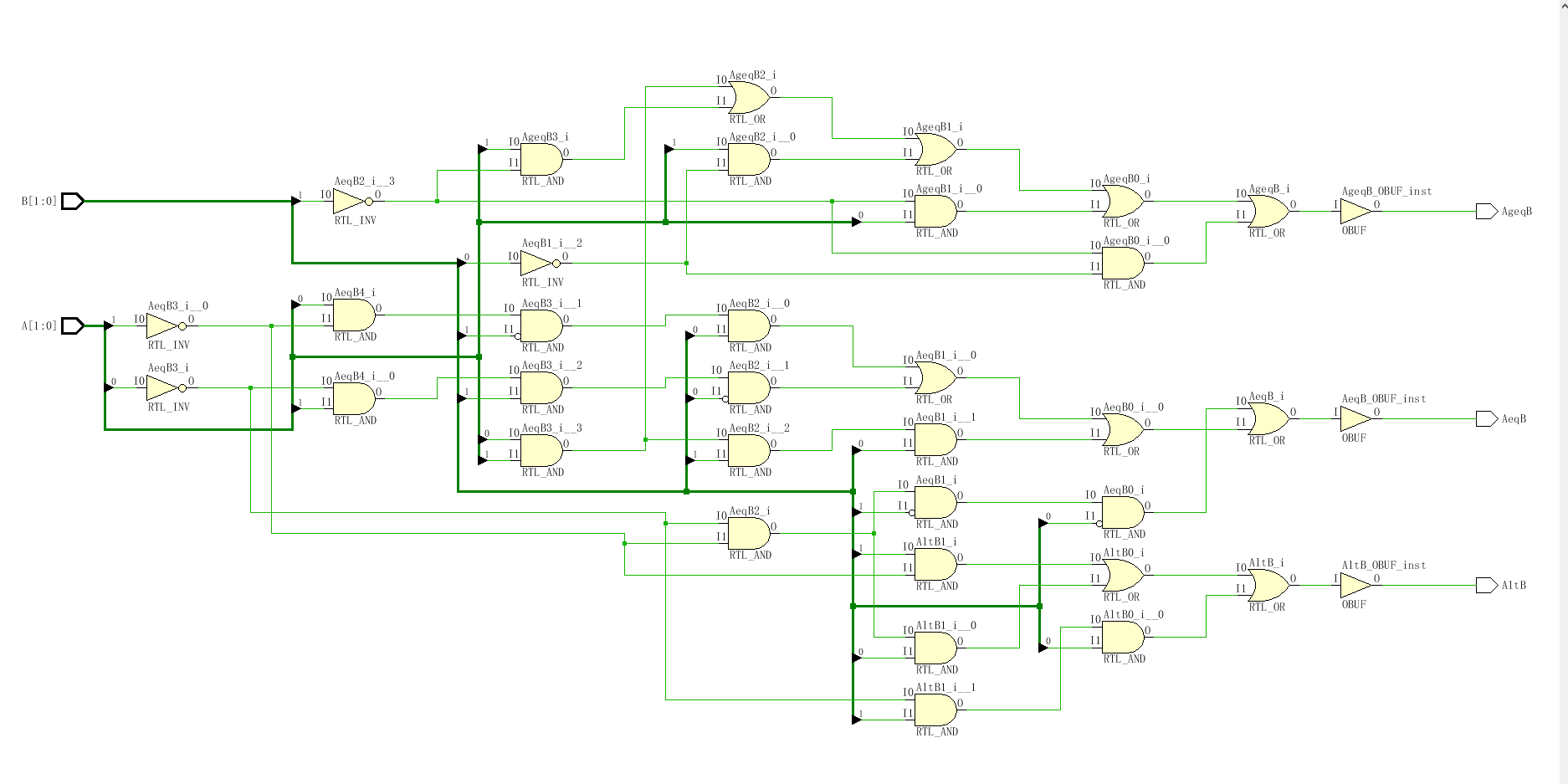
**Test 3**

1. **The main code of project**

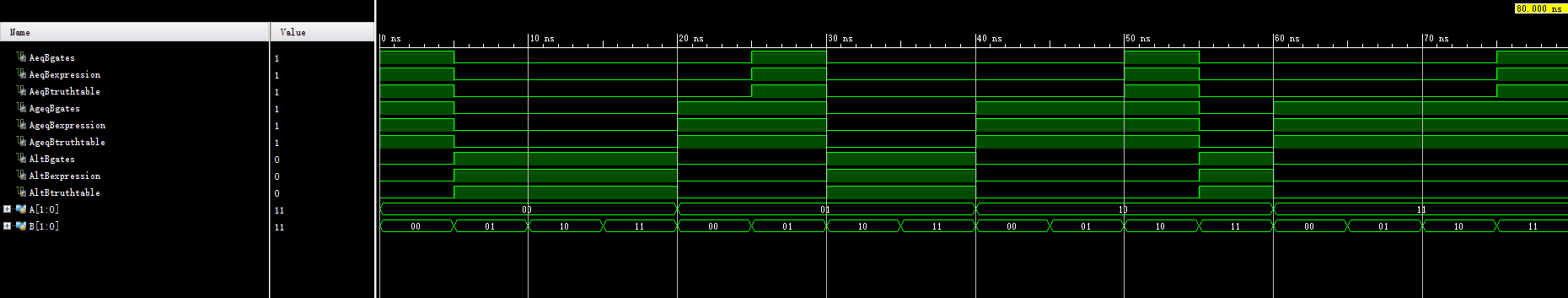
We can get the expression which I have written on handout from the K-maps. And then we can write code of comparatorExpression module and

comparatorGates module.

1. module comparatorExpression(A, B, AeqB, AgeqB, AltB );
2. input [1:0] A, B;
3. output  AeqB, AgeqB, AltB;
5. //assign AeqB = (A === B);
6. //assign AgeqB = (A >= B);
7. //assign AltB = (A < B);
9. assign AeqB = (~A[0] && ~A[1] && ~B[1] && ~B[0]) || (A[0] && ~A[1] && ~B[1] && B[0] ||  (~A[0] && A[1] && B[1] && ~B[0]) || (A[0] && A[1] && B[1] && B[0]));
10. assign AgeqB = (A[0] && A[1]) || (A[1] && ~B[1]) ||  (A[1] && ~B[0]) || (~B[1] && A[0]) || (~B[1] && ~B[0]);
11. assign AltB = (B[1] && ~A[1]) || (~A[1] && ~A[0] && B[0]) ||  (~A[0] && B[1] && B[0]);
13. endmodule
14. module comparatorGates(A, B, AeqB, AgeqB, AltB );
15. input [1:0] A, B;
16. output  AeqB, AgeqB, AltB;
17. wire aeqU1, aeqU2, aeqU3, aeqU4;
18. wire agaeqU1, agaeqU2, agaeqU3, agaeqU4, ageqU5;
19. wire altU1, altU2, altU3;
21. **and** aeqU1and(aeqU1,~A[0], ~A[1], ~B[1], ~B[0]);
22. **and** aeqU2and(aeqU2,A[0], ~A[1], ~B[1], B[0]);
23. **and** aeqU3and(aeqU3,~A[0], A[1], B[1], ~B[0]);
24. **and** aeqU4and(aeqU4,A[0], A[1], B[1], B[0]);
25. **or** aeqOR(AeqB, aeqU1, aeqU2, aeqU3, aeqU4);
27. **and** agaeqU1and(agaeqU1,~B[0], ~B[1]);
28. **and** agaeqU2and(agaeqU2,~B[1], A[1]);
29. **and** agaeqU3and(agaeqU3,A[1], A[0]);
30. **and** agaeqU4and(agaeqU4,A[1], ~B[0]);
31. **and** ageqU5and(ageqU5,~B[1], A[0]);
32. **or** ageqOR(AgeqB, agaeqU1, agaeqU2, agaeqU3, agaeqU4, ageqU5);
34. **and** altU1and(altU1,B[1], ~A[1]);
35. **and** altU2and(altU2, ~A[1], ~A[0], B[0]);
36. **and** altU3and(altU3,~A[0], B[1], B[0]);
37. **or** altOR(AltB,altU1, altU2, altU3);
39. endmodule
40. **The RTL of project**



1. **The simulation of project**



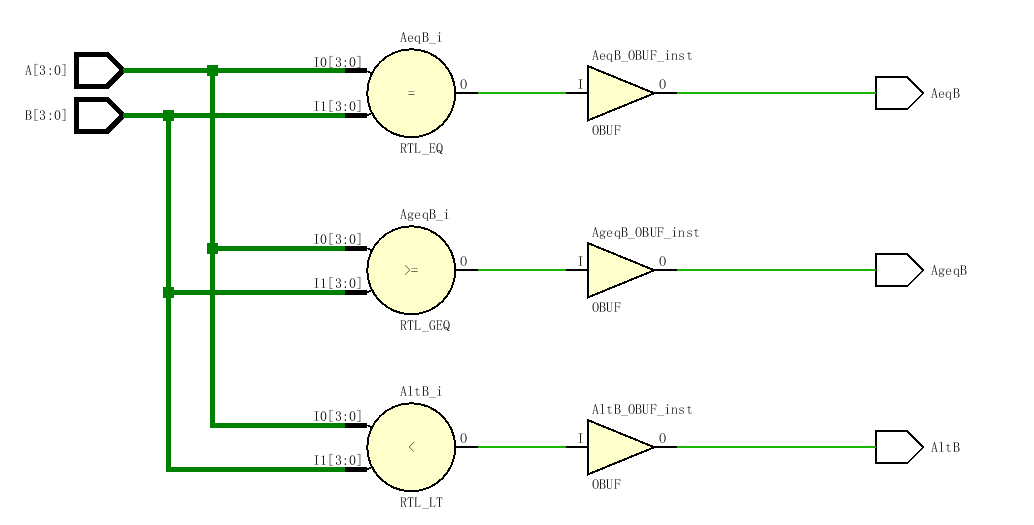
**Test 4**

**a. The main code of project**

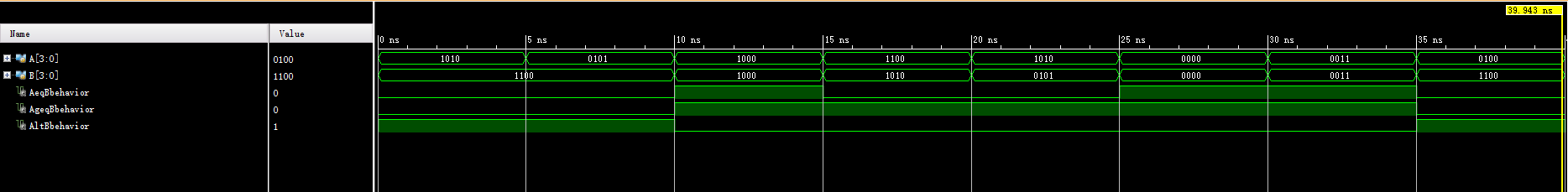
We can easily write the comparator4bit module below through reading test heading.

1. module comparator4bit(A, B, AeqB, AgeqB, AltB );
2. input [3:0] A, B;
3. output  AeqB, AgeqB, AltB;
5. assign AeqB = (A == B);
6. assign AgeqB = (A >= B);
7. assign AltB = (A < B);
9. endmodule
10. module comparitor4bits\_TB;
12. reg[3:0] A,B;
13. wire AeqBbehavior,AgeqBbehavior,AltBbehavior;
14. comparator4bit UnitBehavior
15. (A,B,AeqBbehavior,AgeqBbehavior,AltBbehavior);
17. initial **begin**
18. A=10;B=12;#5;
19. A=5;B=12;#5;
20. A=8;B=8;#5;
21. A=12;B=10;#5;
22. A=10;B=5;#5;
23. A=0;B=0;#5;
24. A=3;B=3;#5;
25. A=20;B=12;#5;
26. $stop;
27. **end**
28. endmodule

**b. The RTL of project**



**c. The simulation of project**



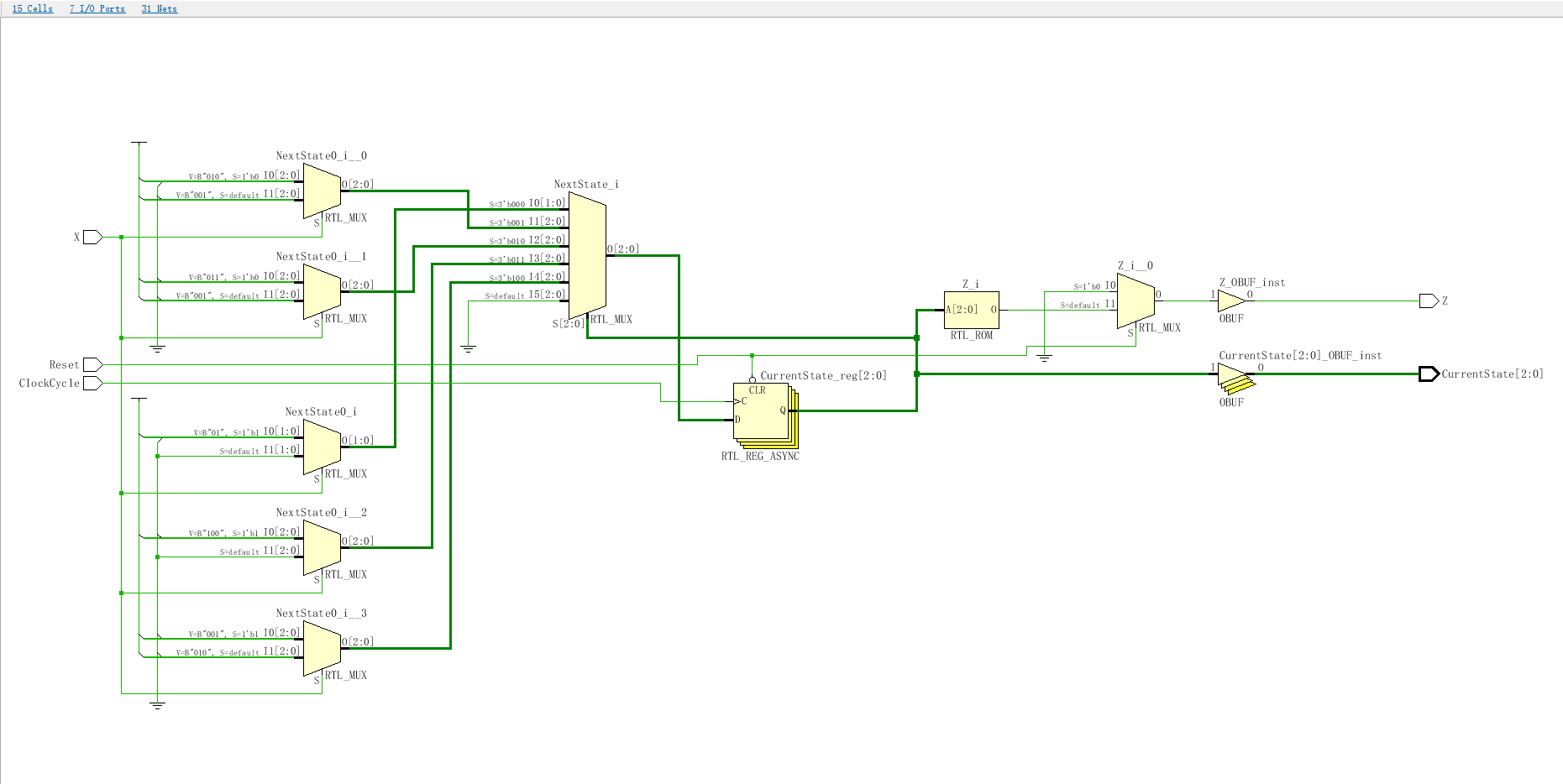
**Test 8**

**a. The main code of project**

We can know the state machine by the heading and the sample timing sequence, and then we could write a state machine module. Also from that the RTL graph and simulation will come out.

1. module statemachine(X, Z, CurrentState, Reset, ClockCycle);
2. input X, Reset, ClockCycle;
3. output reg Z;
4. output reg [2:0] CurrentState;
5. reg [2:0] NextState;
7. parameter Initial = 3'b000, S1 = 3'b001, S10 = 3'b010, S100 = 3'b011, S1001 = 3'b100;
9. always @(negedge Reset **or** posedge  ClockCycle)**begin**
10. **if**(Reset == 0)
11. CurrentState <= Initial;
12. **else**
13. CurrentState <= NextState;
14. **end**
16. always @ (Reset **or** CurrentState)**begin**
17. **if**(Reset == 0)
18. Z = 0;
19. **else**
20. **case**(CurrentState)
21. Initial: Z = 0;
22. S1: Z = 0;
23. S10: Z = 0;
24. S100: Z = 0;
25. S1001: Z = 1;
26. default: Z = 0;
27. endcase
28. **end**
30. always @(X **or** CurrentState)
31. **begin**
32. **case** (CurrentState)
33. Initial:    NextState = (X==1)?S1:Initial;
34. S1:    NextState = (X==0)?S10:S1;
35. S10:    NextState = (X==0)?S100:S1;
36. S100:    NextState = (X==1)?S1001:Initial;
37. S1001:    NextState = (X==1)?S1:S10;
38. default:    NextState = Initial;
39. endcase
40. **end**
42. endmodule

**b.The RTL of project**



**c. The simulation of project**

