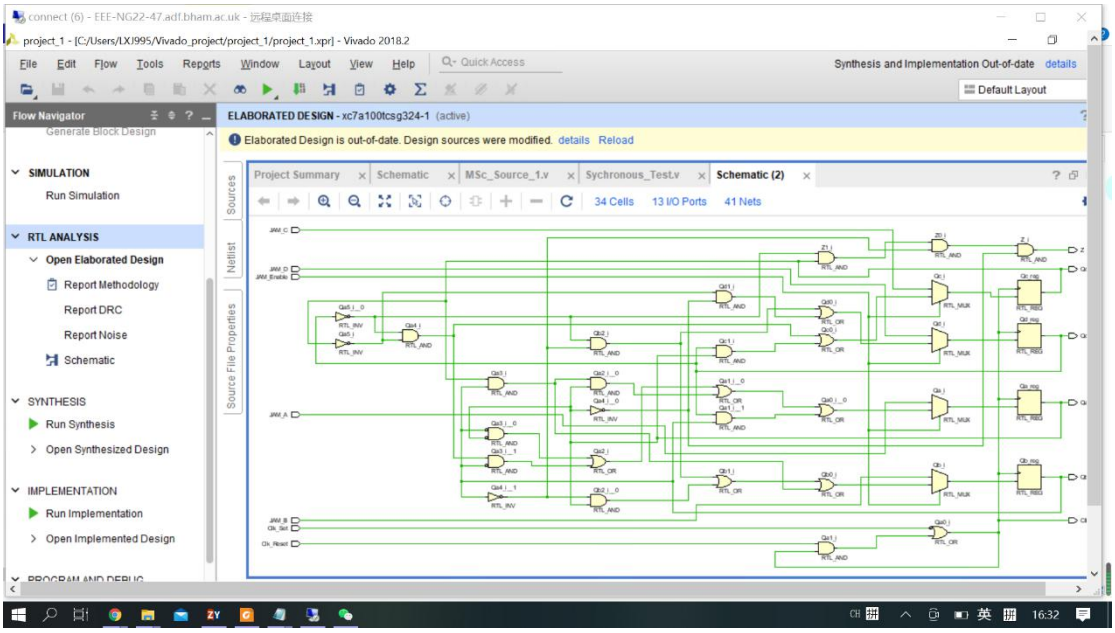
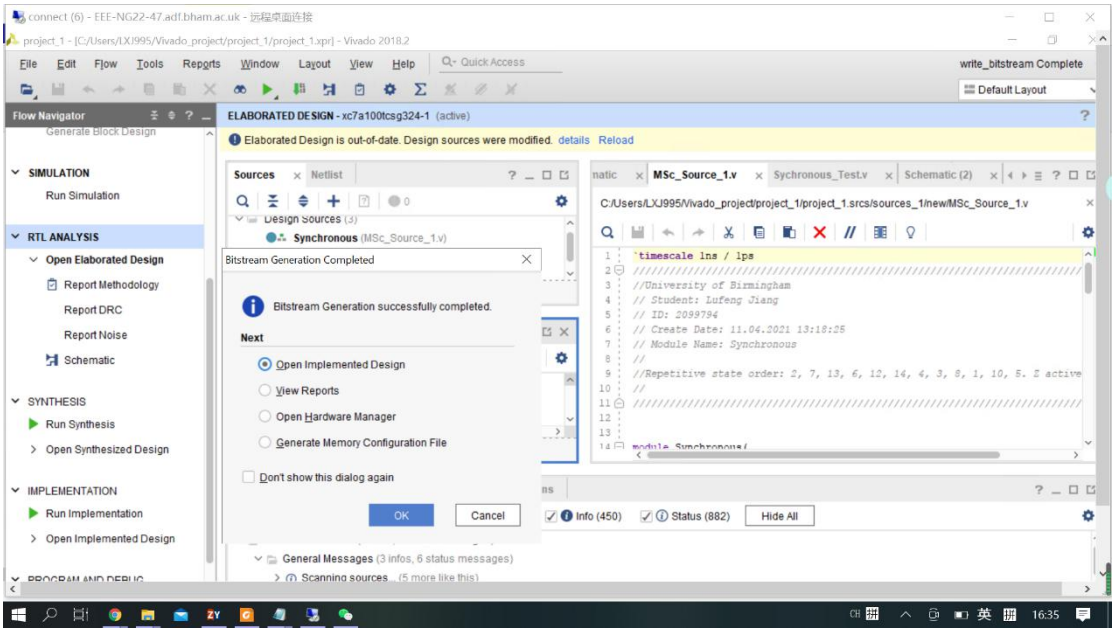
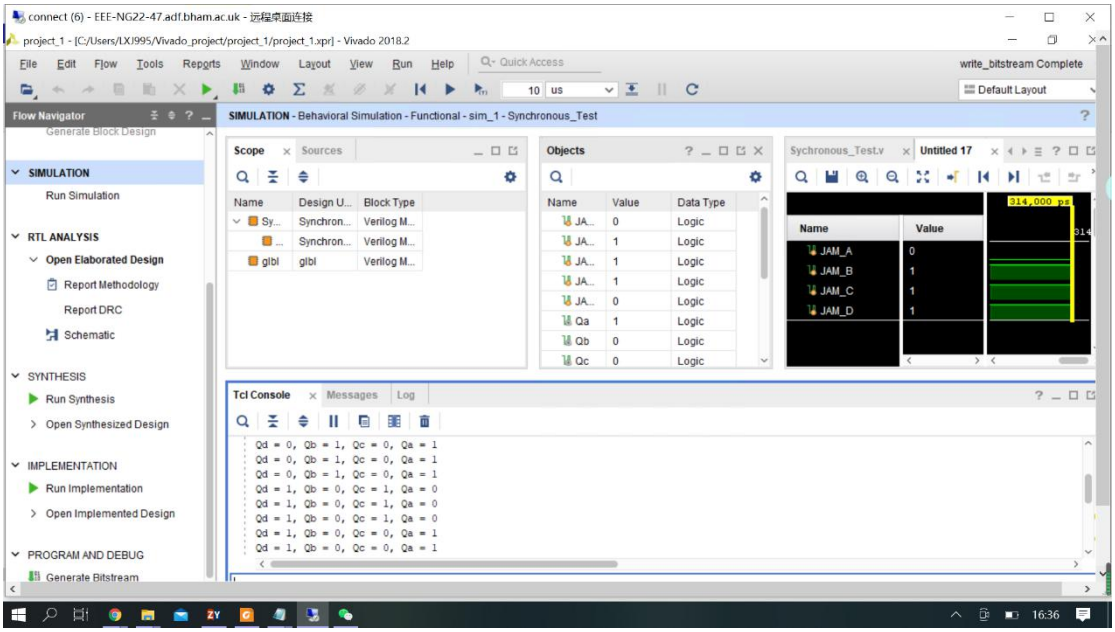
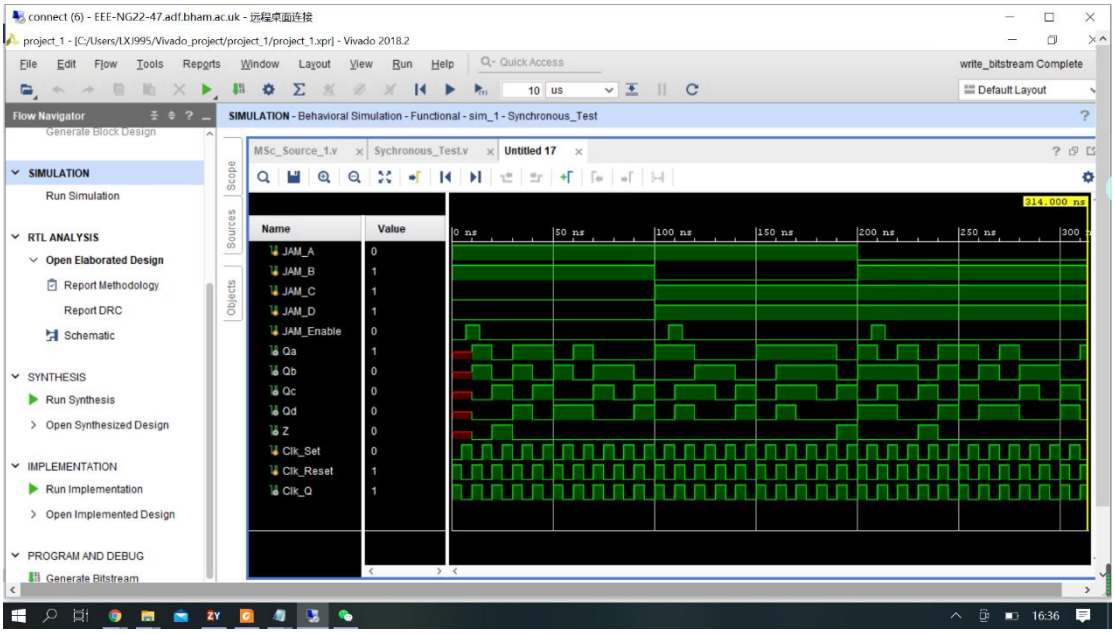
# Synchronous FSM

- Bitstream Generated and simulation






- Verilog Code

(Design)

**Synchronous.v**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//University of Birmingham
// Student: Lufeng Jiang
// ID: 2099794
// Create Date: 11.04.2021 13:18:25
// Module Name: Synchronous
//
//Repetitive state order: 2, 7, 13, 6, 12, 14, 4, 3, 8, 1, 10, 5. Z active on state: 7.
```

```
//
//////////////////////////////////////////////////////////////////////////////////


module Synchronous(
	input JAM_A,
	input JAM_B,
	input JAM_C,
	input JAM_D,
	input JAM_Enable,
	input Clk_Set,
	input Clk_Reset,
	output reg Qa,
	output reg Qb,
	output reg Qc,
	output reg Qd,
	output Clk_Q,
	output Z
	);

	//clock
	reg Clk;

	//Z active at state 7, Code: 0100
	assign Z = ~Qd & Qc & ~Qb & ~Qa;

	//De-bounced clock circuit
	always @(Clk_Set or Clk_Reset)
		begin
			if( (Clk_Set | Clk_Reset)    == 1 )
				begin
					Clk = ~Clk_Set | ( Clk_Reset & Clk );
				end
		end
	assign Clk_Q = Clk;


	always @(posedge Clk)
		begin
			// input lines JAM_A to JAM_D to be clocked onto the output pins
			if (JAM_Enable == 1)
				begin
					Qa <= JAM_A;
					Qb <= JAM_B;
```

```
                    Qc <= JAM_C;
                    Qd <= JAM_D;
                end

            else
                //Transition equations
                begin
                    Qa <= (~Qb & ~Qa) | (~Qd & ~Qc & ~Qb) | (Qc & Qb & Qa) |
(Qd & Qb);
                    Qb <= (~Qd & ~Qb) | (~Qb & Qa) | (Qc & Qb);
                    Qc <= (~Qd & ~Qc) | (Qd & Qa);
                    Qd <= (~Qd & ~Qb) | (~Qc & ~Qa);
                end
        end
endmodule
```

(Test)

**Synchronous_Test.v**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//University of Birmingham
// Student: Lufeng Jiang
// ID: 2099794
// Create Date: 11.04.2021 13:18:25
// Module Name: Synchronous_Test

//Repetitive state order: 2, 7, 13, 6, 12, 14, 4, 3, 8, 1, 10, 5. Z active on state: 7.
//
//////////////////////////////////////////////////////////////////////////////////


module Synchronous_Test();
    reg JAM_A;
    reg JAM_B;
    reg JAM_C;
    reg JAM_D;
    reg JAM_Enable;

    wire Qa;
    wire Qb;
    wire Qc;
    wire Qd;
    wire Z;
```

```
    reg Clk_Set;
    reg Clk_Reset;
    wire Clk_Q;

    //generate clock
    initial begin
        Clk_Set = 0;
        Clk_Reset = 1;
        forever
            begin
                #5 Clk_Set <= ~Clk_Set;
                  Clk_Reset <= ~Clk_Reset;
            end
    end

    initial begin
    $display("Start from state 2, repetive state order: 2, 7, 13, 6, 12, 14, 4, 3, 8, 1, 10, 5.");
    $monitor ("JAM_Enable = %b, Qd = %b, Qb = %b, Qc = %b, Qa = %b",JAM_Enable, Qd, Qc, Qb, Qa);
    // initialization of States, start from state 2
        JAM_Enable <= 0;
        JAM_A <= 1;
        JAM_B <= 1;
        JAM_C <= 0;
        JAM_D <= 0;
        #7 JAM_Enable <= 1;
        #7 JAM_Enable <= 0;
        #300;
        $finish;
    end

    initial begin
    $display("After 100ns, JAM_Enable=1, not allowed state 9(code:1101) should transit to allowed state 4(code: 0110)");
        // after 100ns, force into not allowed state 9(code:1101), will transit to allowed state 4(code: 0110)
        #100;
        JAM_Enable <= 0;
        JAM_A <= 1;
        JAM_B <= 0;
        JAM_C <= 1;
        JAM_D <= 1;
```

```
        #7 JAM_Enable <= 1;
        #7 JAM_Enable <= 0;
        #200;
        $finish;
    end

    initial begin
    $display("After 200ns, JAM_Enable=1, not allowed state 11(code:1110) should transit to allowed state 2(code: 0011)");
        // after 200ns, force into not allowed state 11(code:1110), will transit to allowed state 2(code: 0011)
        #200;
        JAM_Enable <= 0;
        JAM_A <= 0;
        JAM_B <= 1;
        JAM_C <= 1;
        JAM_D <= 1;
        #7 JAM_Enable <= 1;
        #7 JAM_Enable <= 0;
        #100;
        $finish;
        end



    Synchronous PhilModule(
        .Clk_Set(Clk_Set), .Clk_Reset(Clk_Reset), .Clk_Q(Clk_Q), .JAM_A(JAM_A), .JAM_B(JAM_B), .JAM_C(JAM_C), .JAM_D(JAM_D), .JAM_Enable(JAM_Enable),
        .Qa(Qa), .Qb(Qb), .Qc(Qc), .Qd(Qd), .Z(Z));

endmodule
```

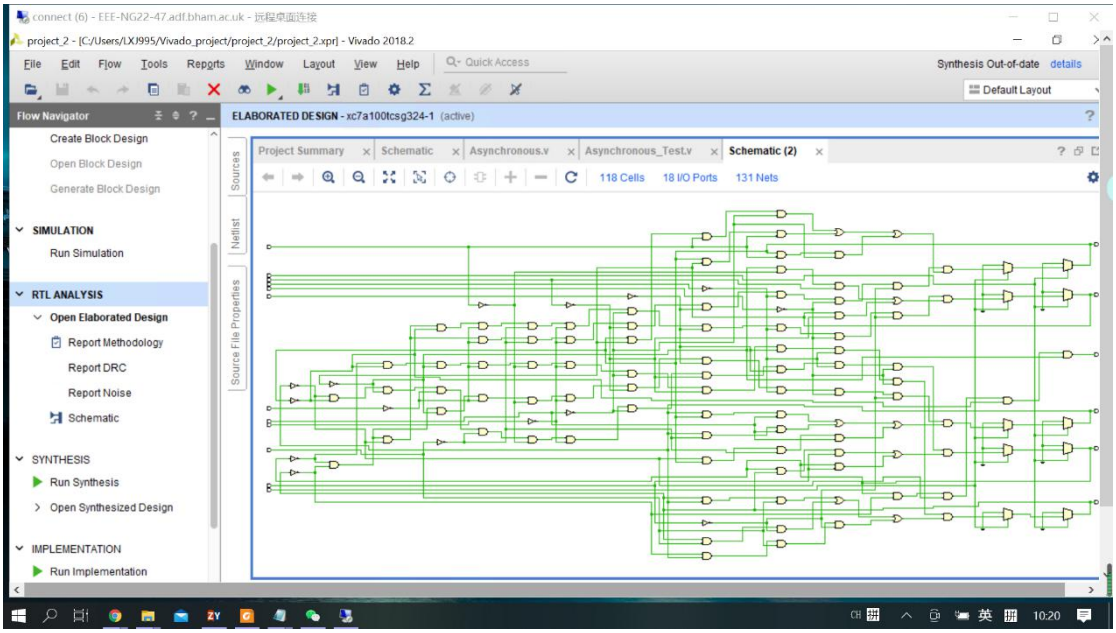
output of **'$monitor' and '$display'**

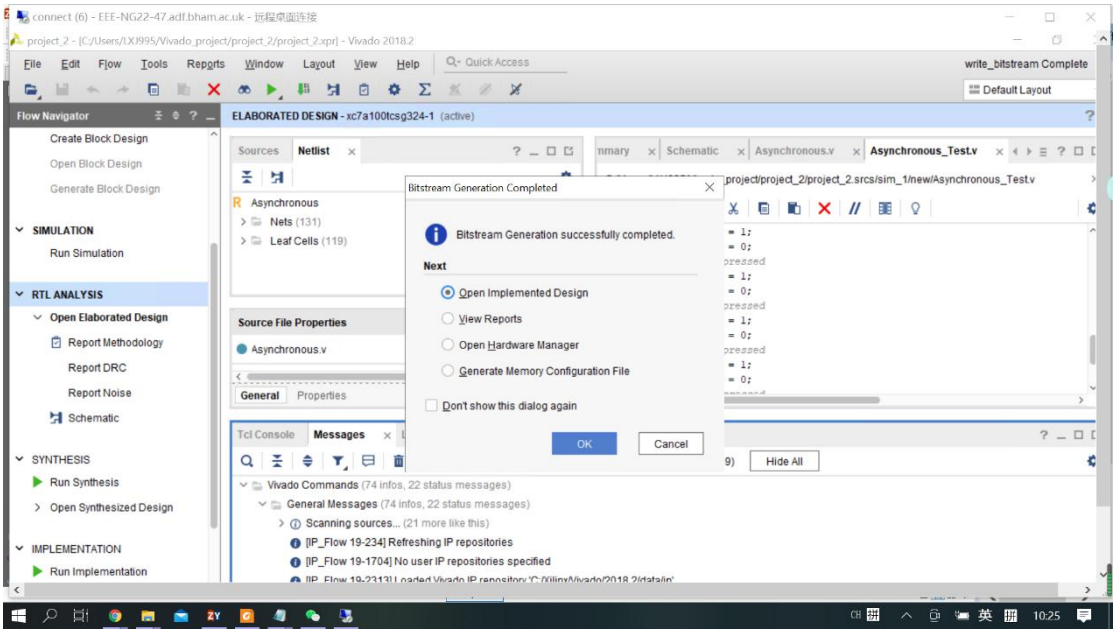
```
Start from state 2, repetive state order: 2, 7, 13, 6, 12, 14, 4, 3, 8, 1, 10, 5.
After 100ns, JAM_Enable=1, not allowed state 9(code:1101) should transit to allowed state 4(code: 0110)
After 200ns, JAM_Enable=1, not allowed state 11(code:1110) should transit to allowed state 2(code: 0011)
JAM_Enable = 0, Qd = x, Qb = x, Qc = x, Qa = x
JAM_Enable = 1, Qd = x, Qb = x, Qc = x, Qa = x
JAM_Enable = 1, Qd = 0, Qb = 0, Qc = 1, Qa = 1
JAM_Enable = 0, Qd = 0, Qb = 0, Qc = 1, Qa = 1
```

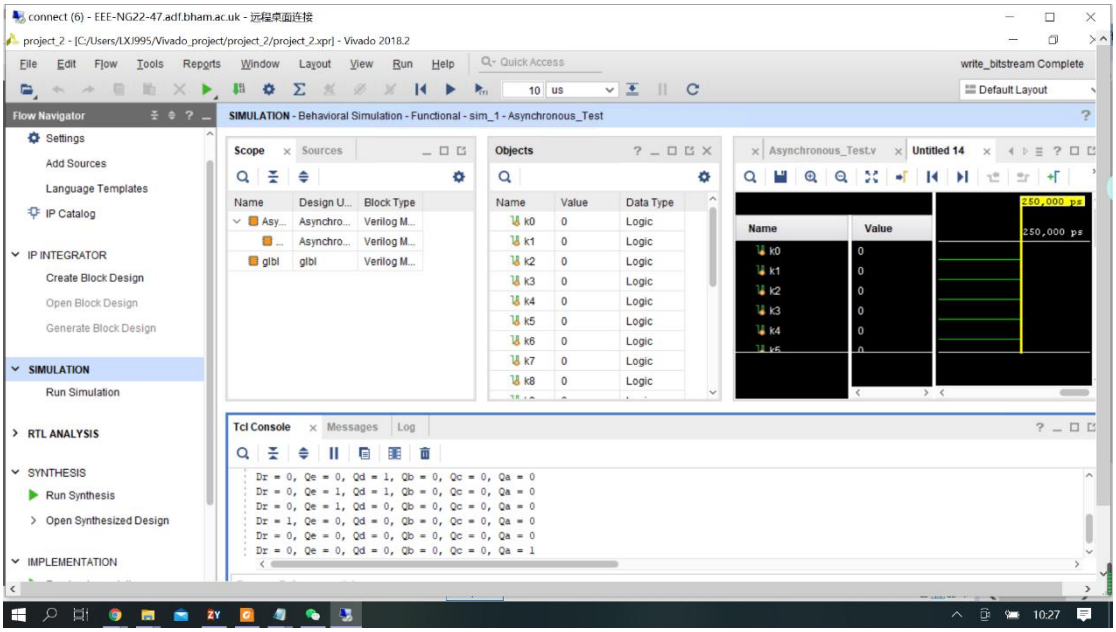
```
JAM_Enable = 0, Qd = 0, Qb = 1, Qc = 0, Qa = 0
JAM_Enable = 0, Qd = 1, Qb = 0, Qc = 1, Qa = 1
JAM_Enable = 0, Qd = 0, Qb = 1, Qc = 0, Qa = 1
JAM_Enable = 0, Qd = 1, Qb = 0, Qc = 1, Qa = 0
JAM_Enable = 0, Qd = 1, Qb = 0, Qc = 0, Qa = 1
JAM_Enable = 0, Qd = 0, Qb = 1, Qc = 1, Qa = 0
JAM_Enable = 0, Qd = 0, Qb = 0, Qc = 1, Qa = 0
JAM_Enable = 0, Qd = 1, Qb = 1, Qc = 0, Qa = 0
JAM_Enable = 0, Qd = 0, Qb = 0, Qc = 0, Qa = 1
JAM_Enable = 1, Qd = 0, Qb = 0, Qc = 0, Qa = 1
JAM_Enable = 1, Qd = 1, Qb = 1, Qc = 0, Qa = 1
JAM_Enable = 0, Qd = 1, Qb = 1, Qc = 0, Qa = 1
JAM_Enable = 0, Qd = 0, Qb = 1, Qc = 1, Qa = 0
JAM_Enable = 0, Qd = 0, Qb = 0, Qc = 1, Qa = 0
JAM_Enable = 0, Qd = 1, Qb = 1, Qc = 0, Qa = 0
JAM_Enable = 0, Qd = 0, Qb = 0, Qc = 0, Qa = 1
JAM_Enable = 0, Qd = 1, Qb = 1, Qc = 1, Qa = 1
JAM_Enable = 0, Qd = 0, Qb = 1, Qc = 1, Qa = 1
JAM_Enable = 0, Qd = 0, Qb = 0, Qc = 1, Qa = 1
JAM_Enable = 0, Qd = 0, Qb = 1, Qc = 0, Qa = 0
JAM_Enable = 0, Qd = 1, Qb = 0, Qc = 1, Qa = 1
JAM_Enable = 1, Qd = 1, Qb = 0, Qc = 1, Qa = 1
JAM_Enable = 1, Qd = 1, Qb = 1, Qc = 1, Qa = 0
JAM_Enable = 0, Qd = 1, Qb = 1, Qc = 1, Qa = 0
JAM_Enable = 0, Qd = 0, Qb = 0, Qc = 1, Qa = 1
JAM_Enable = 0, Qd = 0, Qb = 1, Qc = 0, Qa = 0
JAM_Enable = 0, Qd = 1, Qb = 0, Qc = 1, Qa = 1
JAM_Enable = 0, Qd = 0, Qb = 1, Qc = 0, Qa = 1
JAM_Enable = 0, Qd = 1, Qb = 0, Qc = 1, Qa = 0
JAM_Enable = 0, Qd = 1, Qb = 0, Qc = 0, Qa = 1
JAM_Enable = 0, Qd = 0, Qb = 1, Qc = 1, Qa = 0
JAM_Enable = 0, Qd = 0, Qb = 0, Qc = 1, Qa = 0
JAM_Enable = 0, Qd = 1, Qb = 1, Qc = 0, Qa = 0
JAM_Enable = 0, Qd = 0, Qb = 0, Qc = 0, Qa = 1
$finish called at time : 314 ns : File "C:/Users/LXJ995/Vivado_project/project_1/project_1.srcs/sources_1/new/Sychronous_Test.v" Line 54
```

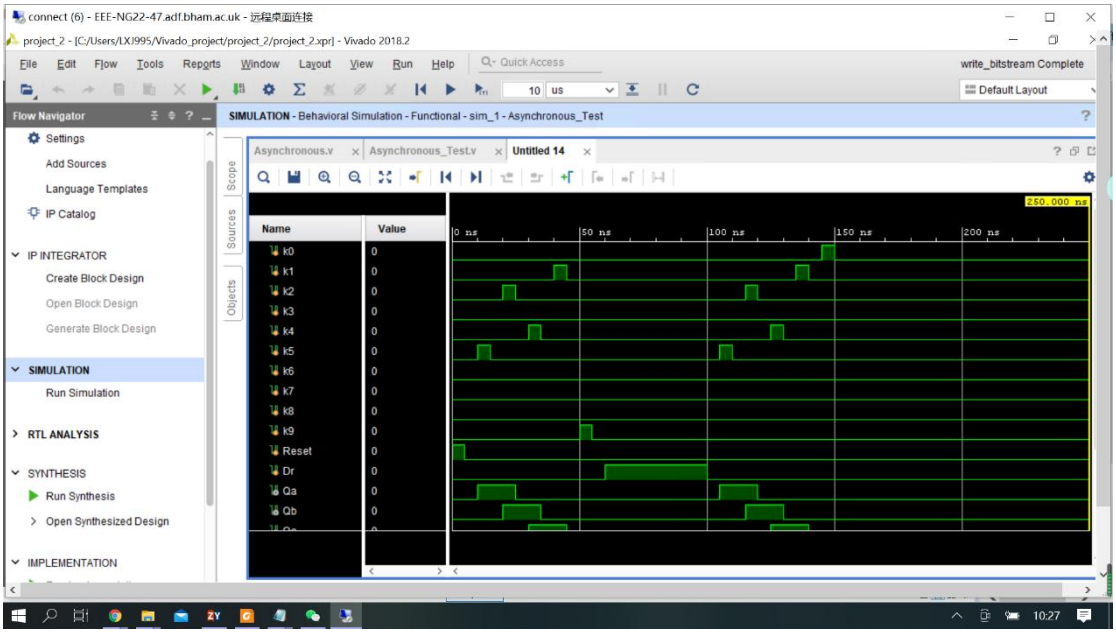
## Asynchronous FSM

- Bitstream Generated and simulation






- Verilog Code

(Design)

**Asynchronous.v**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//University of Birmingham
// Student: Lufeng Jiang
// ID: 2099794
// Create Date: 11.04.2021 13:18:25
// Module Name: Asynchronous
```

```
//Lock sequence: 5,2,4,1,9.
//
//////////////////////////////////////////////////////////////////////////////////

module Asynchronous(
	input k0,
	input k1,
	input k2,
	input k3,
	input k4,
	input k5,
	input k6,
	input k7,
	input k8,
	input k9,
	input Dr,
	input Reset,
	output reg Qa,
	output reg Qb,
	output reg Qc,
	output reg Qd,
	output reg Qe,
	output Z
	);

	//Z active	when five-digit code entered correctly
	assign Z = Qe & ~Qd & ~Qc & ~Qb & ~Qa;

	//When any key pressed
	always @( k0 or k1 or k2 or k3 or k4 or k5 or k6 or k7 or k8 or k9	or Reset or
Dr )
		begin
			//Reset to idling state A(00000)
			if ( Reset == 1)
				begin
					Qa <= 0;
					Qb <= 0;
					Qc <= 0;
					Qd <= 0;
					Qe <= 0;
				end

			//Door open switch causing a return to the idling state
```

```
                else if ( Dr == 1 )
                        begin
                                Qa <= 0;
                                Qb <= 0;
                                Qc <= 0;
                                Qd <= 0;
                                Qe <= 0;
                        end

                else
                        begin
                        //Traisition equations
                                Qa <= (~Qe & ~Qd & ~Qc & ~Qb & ~k0 & ~k1 & ~k3 & ~k4 & ~k6 & ~k7 & ~k8 & ~k9) & ( (~Qa & ~k2 & k5) | (Qa & ~k5) );
                                Qb <= (~Qe & ~Qd & ~Qc & ~k0 & ~k1 & ~k3 & ~k5 & ~k6 & ~k7 & ~k8 & ~k9) & ( (~Qb & Qa & k2 & ~k4) | (Qb & ~Qa & ~k2) | ( Qb & ~k2 & ~k4) );
                                Qc <= (~Qe & ~Qd & ~Qa & ~k0 & ~k2 & ~k3 & ~k5 & ~k6 & ~k7 & ~k8 & ~k9) & ( (Qc & ~Qb & ~k4) | (Qc & ~k1 & ~k4) | ( ~Qc & Qb & ~k1 & k4) );
                                Qd <= (~Qe & ~Qb & ~Qa & ~k0 & ~k2 & ~k3 & ~k4 & ~k5 & ~k6 & ~k7 & ~k8) & ( (Qd & ~Qc & ~k1 ) | (Qd & ~k1 & ~k9) | (~Qd & Qc & k1 & ~k9 ) );
                                Qe <= (~Qc & ~Qb & ~Qa & ~k0 & ~k1 & ~k2 & ~k3 & ~k4 & ~k5 & ~k6 & ~k7 & ~k8) & ( (~Qe & Qd & k9) | (Qe & ~k9) );
                        end
            end
endmodule
```

(Test)

**Asynchronous.v**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//University of Birmingham
// Student: Lufeng Jiang
// ID: 2099794
// Create Date: 11.04.2021 13:18:25
// Module Name: Asynchronous_Test

//Lock sequence: 5,2,4,1,9.
//
//////////////////////////////////////////////////////////////////////////////////
```

```
module Asynchronous_Test();
	reg k0;
	reg k1;
	reg k2;
	reg k3;
	reg k4;
	reg k5;
	reg k6;
	reg k7;
	reg k8;
	reg k9;
	reg Reset;
	reg Dr;

	wire Qa;
	wire Qb;
	wire Qc;
	wire Qd;
	wire Qe;
	wire Z;

	initial begin
	$display("correct five-digit code entered(5,2,4,1,9) and door open, state order: ABGCHDIEJFA ");
	$monitor ("Dr = %b, Qe = %b, Qd = %b, Qb = %b, Qc = %b, Qa = %b",Dr, Qe, Qd, Qc, Qb, Qa);
	//initialization of inputs( no key pressed )
		k0 <= 0;
		k1 <= 0;
		k2 <= 0;
		k3 <= 0;
		k4 <= 0;
		k5 <= 0;
		k6 <= 0;
		k7 <= 0;
		k8 <= 0;
		k9 <= 0;

		//correct five-digit code entered(5,2,4,1,9) and door open
		Dr = 0 ;
		//Reset
		Reset =1;
```

```
        #5 Reset = 0;
        //k5 pressed
        #5 k5 = 1;
        #5 k5 = 0;
        //k2 pressed
        #5 k2 = 1;
        #5 k2 = 0;
        //k4 pressed
        #5 k4 = 1;
        #5 k4 = 0;
        //k1 pressed
        #5 k1 = 1;
        #5 k1 = 0;
        //k9 pressed
        #5 k9 = 1;
        #5 k9 = 0;
        //Door open switch, return to state A
        #5 Dr = 1 ;
        #200;
        $finish;
    end

    initial begin
    $display("wrong five-digit code entered(5,2,4,1,0), state order: ABGCHDIEA ");
    //after 100ns, close the door and enter wrong five-digit code (5,2,4,1,0)
        #100;
        Dr = 0 ;
        //k5 pressed
        #5 k5 = 1;
        #5 k5 = 0;
        //k2 pressed
        #5 k2 = 1;
        #5 k2 = 0;
        //k4 pressed
        #5 k4 = 1;
        #5 k4 = 0;
        //k1 pressed
        #5 k1 = 1;
        #5 k1 = 0;
        //k0 pressed
        #5 k0 = 1;
        #5 k0 = 0;
        #100;
        $finish;
```

```
    end


    Asynchronous
PhilModule1( .Dr(Dr), .Reset(Reset), .k0(k0), .k1(k1), .k2(k2), .k3(k3), .k4(k4), .k5(k5), .
k6(k6), .k7(k7), .k8(k8),
.k9(k9), .Qa(Qa), .Qb(Qb), .Qc(Qc), .Qd(Qd), .Qe(Qe), .Z(Z) );

endmodule
```

output of **'$monitor' and '$display'**

```
correct five-digit code entered(5,2,4,1,9) and door open, Dr=1,, state order:
ABGCHDIEJFA
Door close, Dr=0, wrong five-digit code entered(5,2,4,1,0), state order: ABGCHDIEA
Dr = 0, Qe = 0, Qd = 0, Qb = 0, Qc = 0, Qa = 0
Dr = 0, Qe = 0, Qd = 0, Qb = 0, Qc = 0, Qa = 1
Dr = 0, Qe = 0, Qd = 0, Qb = 0, Qc = 1, Qa = 1
Dr = 0, Qe = 0, Qd = 0, Qb = 0, Qc = 1, Qa = 0
Dr = 0, Qe = 0, Qd = 0, Qb = 1, Qc = 1, Qa = 0
Dr = 0, Qe = 0, Qd = 0, Qb = 1, Qc = 0, Qa = 0
Dr = 0, Qe = 0, Qd = 1, Qb = 1, Qc = 0, Qa = 0
Dr = 0, Qe = 0, Qd = 1, Qb = 0, Qc = 0, Qa = 0
Dr = 0, Qe = 1, Qd = 1, Qb = 0, Qc = 0, Qa = 0
Dr = 0, Qe = 1, Qd = 0, Qb = 0, Qc = 0, Qa = 0
Dr = 1, Qe = 0, Qd = 0, Qb = 0, Qc = 0, Qa = 0
Dr = 0, Qe = 0, Qd = 0, Qb = 0, Qc = 0, Qa = 0
Dr = 0, Qe = 0, Qd = 0, Qb = 0, Qc = 0, Qa = 1
Dr = 0, Qe = 0, Qd = 0, Qb = 0, Qc = 1, Qa = 1
Dr = 0, Qe = 0, Qd = 0, Qb = 0, Qc = 1, Qa = 0
Dr = 0, Qe = 0, Qd = 0, Qb = 1, Qc = 1, Qa = 0
Dr = 0, Qe = 0, Qd = 0, Qb = 1, Qc = 0, Qa = 0
Dr = 0, Qe = 0, Qd = 1, Qb = 1, Qc = 0, Qa = 0
Dr = 0, Qe = 0, Qd = 1, Qb = 0, Qc = 0, Qa = 0
Dr = 0, Qe = 0, Qd = 0, Qb = 0, Qc = 0, Qa = 0
$finish called at time : 250 ns : File
"C:/Users/LXJ995/Vivado_project/project_2/project_2.srcs/sim_1/new/Asynchronou
s_Test.v" Line 97
```