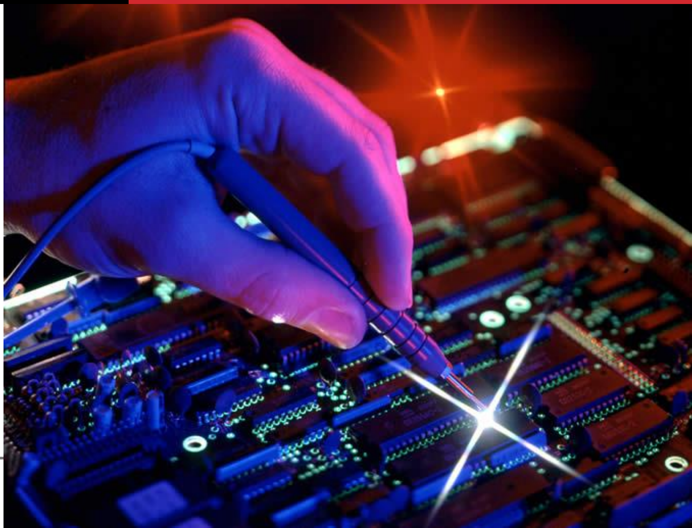
Advanced Computer Architecture & Advanced Microprocessor System

VIVADO DESIGN EXAMPLE

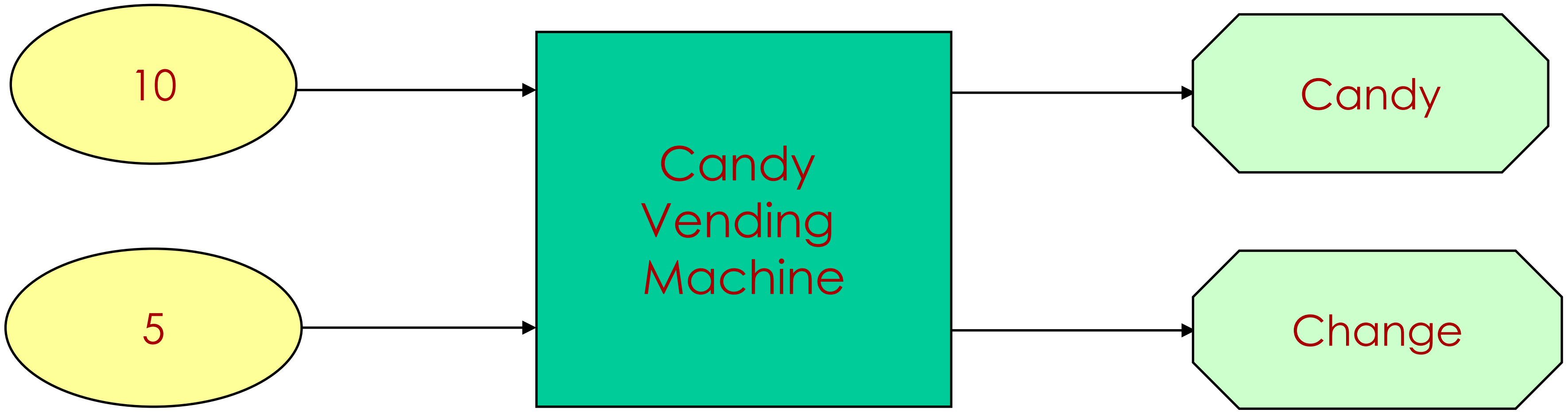
Dennis A. N. Gookyi



CONTENTS

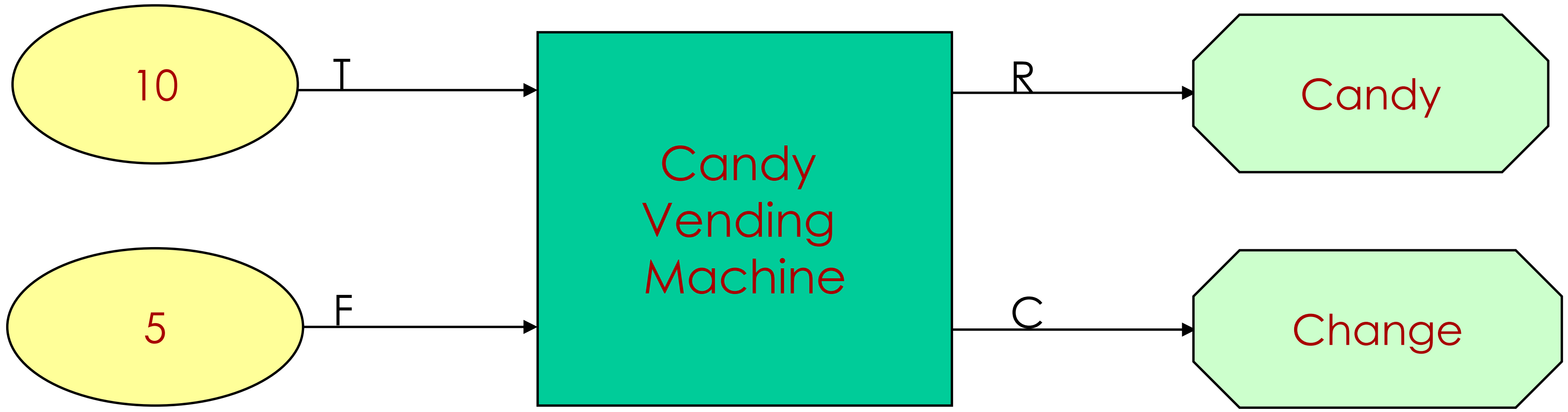
DESIGN EXAMPLE

- Design Specification: Candy Vending Machine
  - One candy cost 20
  - Only 10 and 5 coins are accepted
  - When money exceeding 20 comes in, candy and change are released
    - The money received cannot exceed 25 won



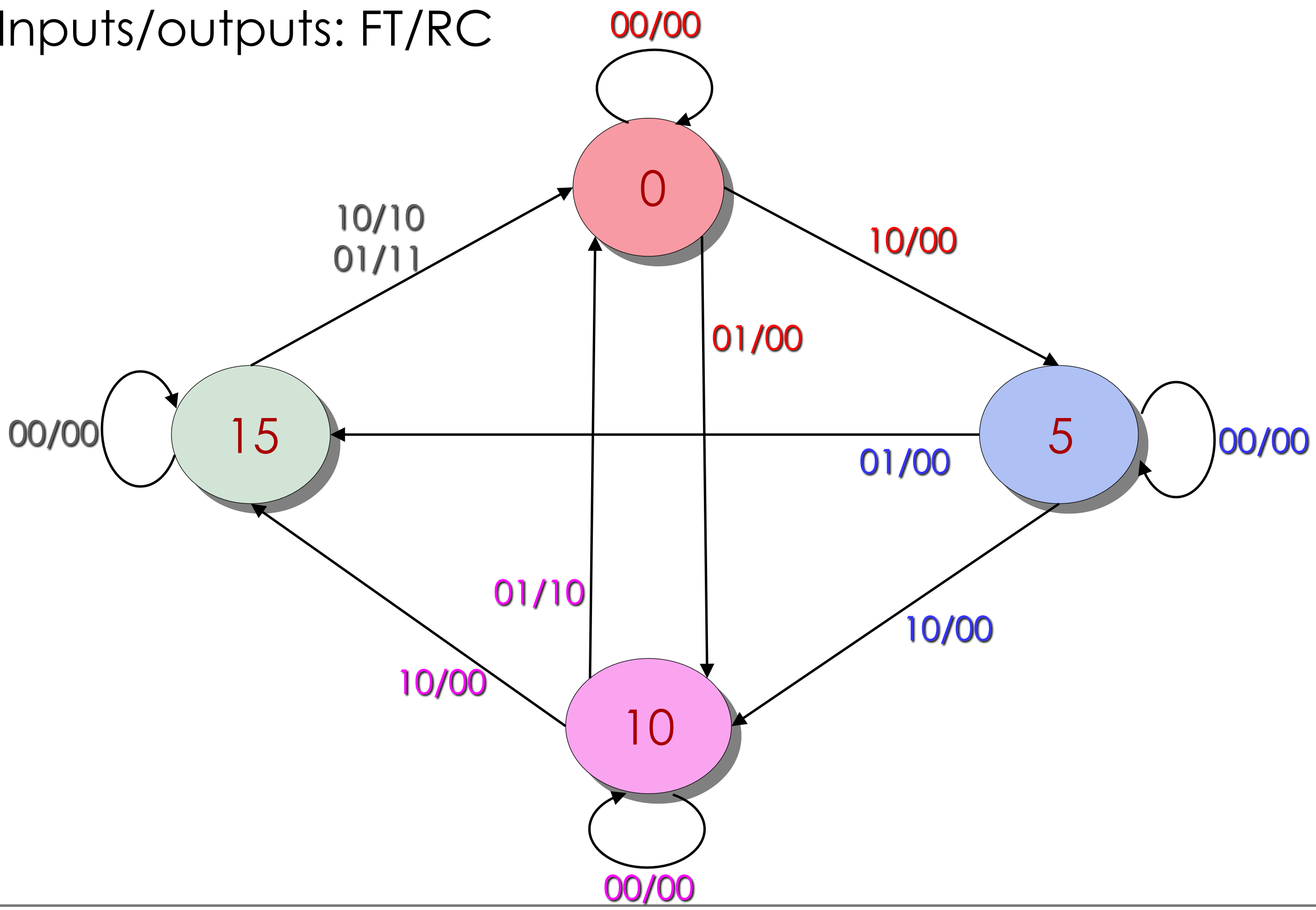
DESIGN EXAMPLE

- Input/Output Specifications
  - Inputs
    - T (Ten): Enter 10 coin
    - F (Five): Enter 5 coin
  - Outputs
    - R (Release): Candy release
    - C (Change): Change release



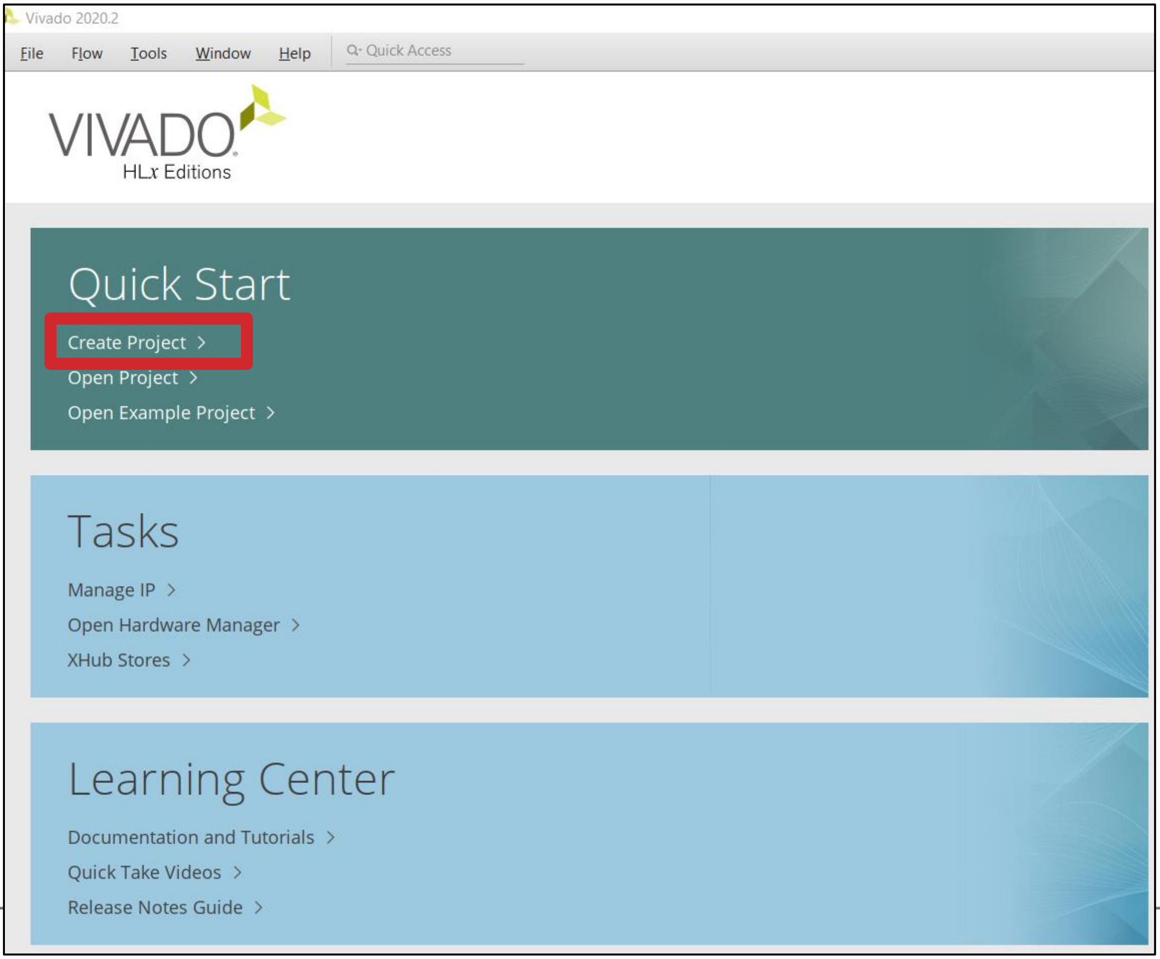
DESIGN EXAMPLE

- Mealy-type State Machine
  - Inputs/outputs: FT/RC



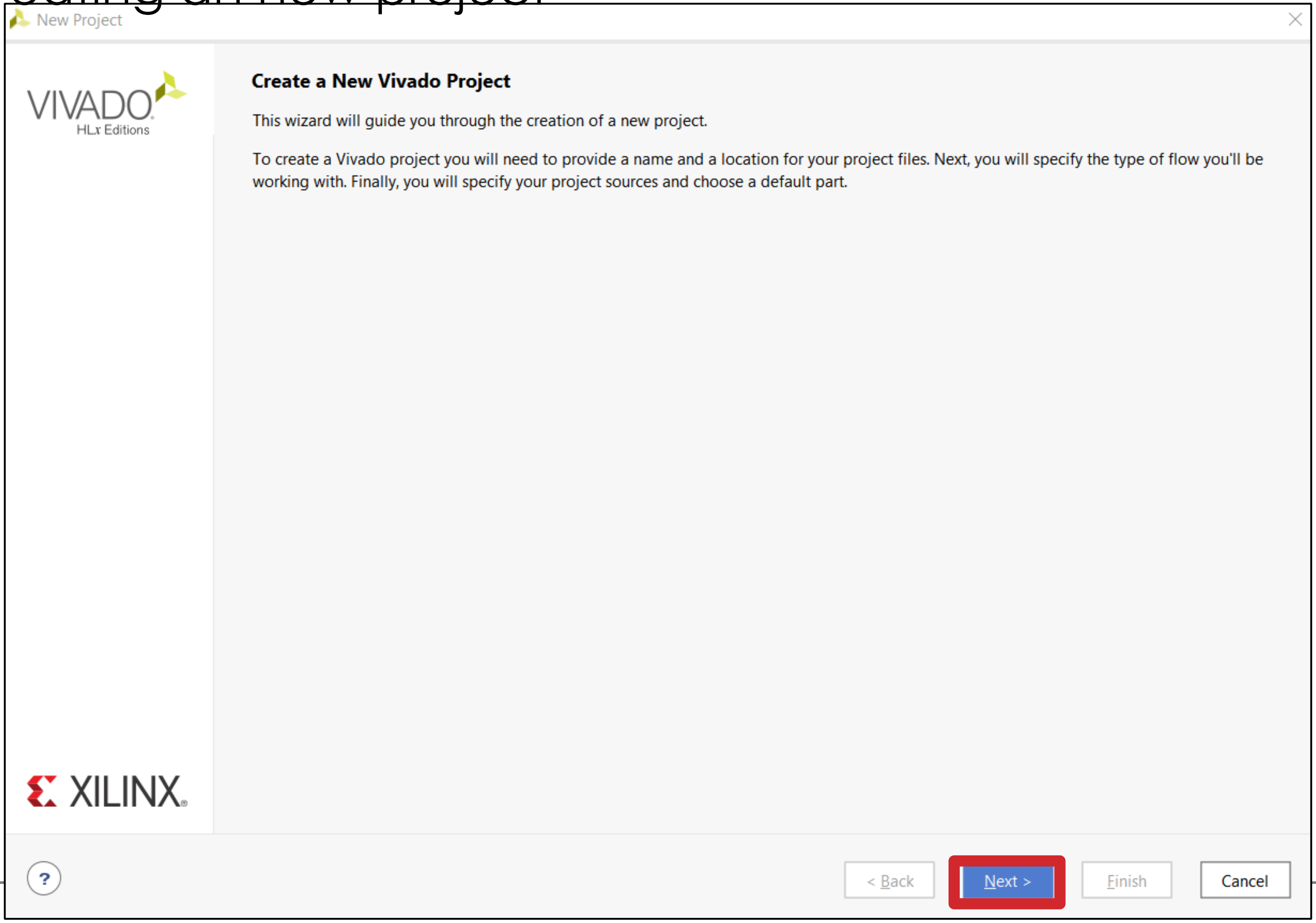
DESIGN EXAMPLE

- Vivado project
  - Creating a new project



DESIGN EXAMPLE

- Vivado project
  - Creating an new project


New Project

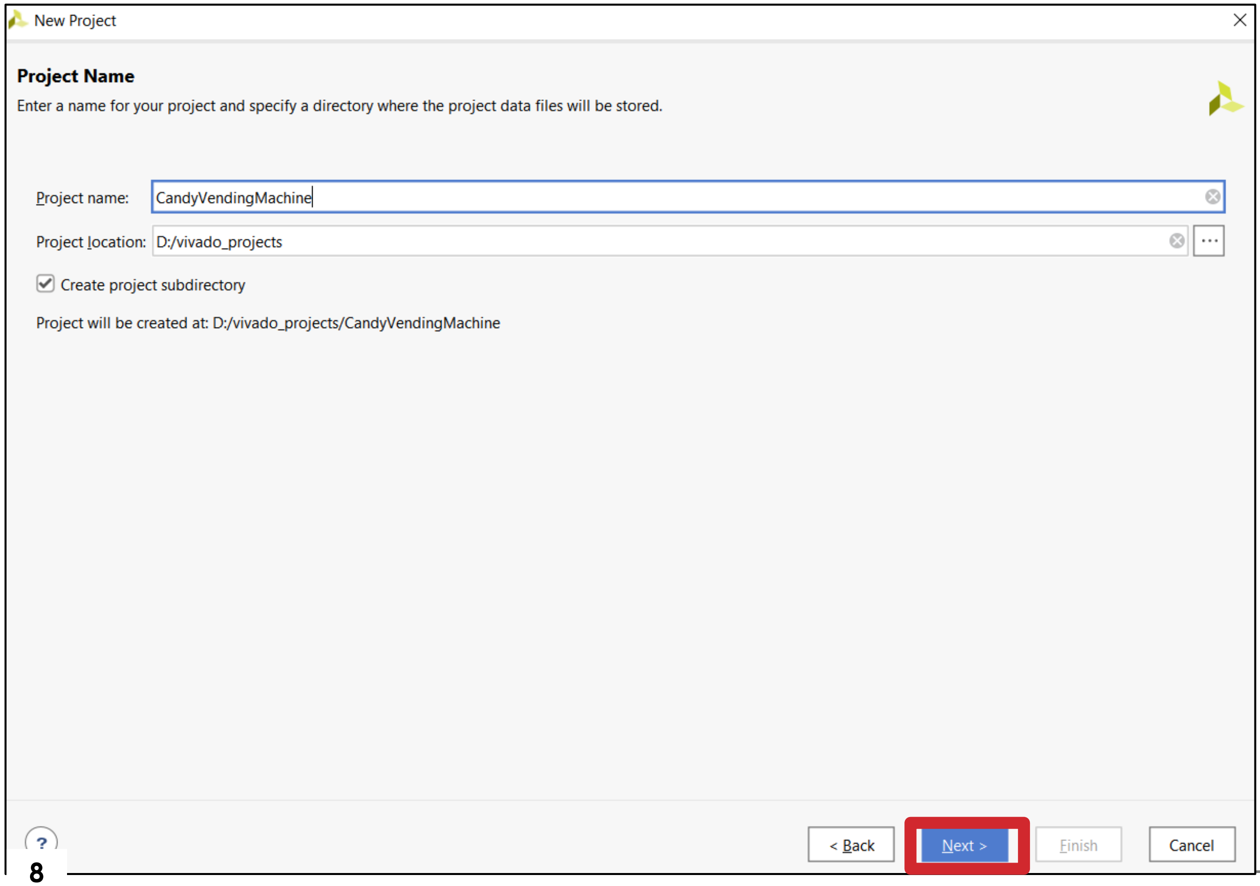
Create a New Vivado Project

This wizard will guide you through the creation of a new project.

To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.

DESIGN EXAMPLE

- Vivado project
  - Creating a new project



New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: CandyVendingMachine

Project location: D:/vivado_projects

☑ Create project subdirectory

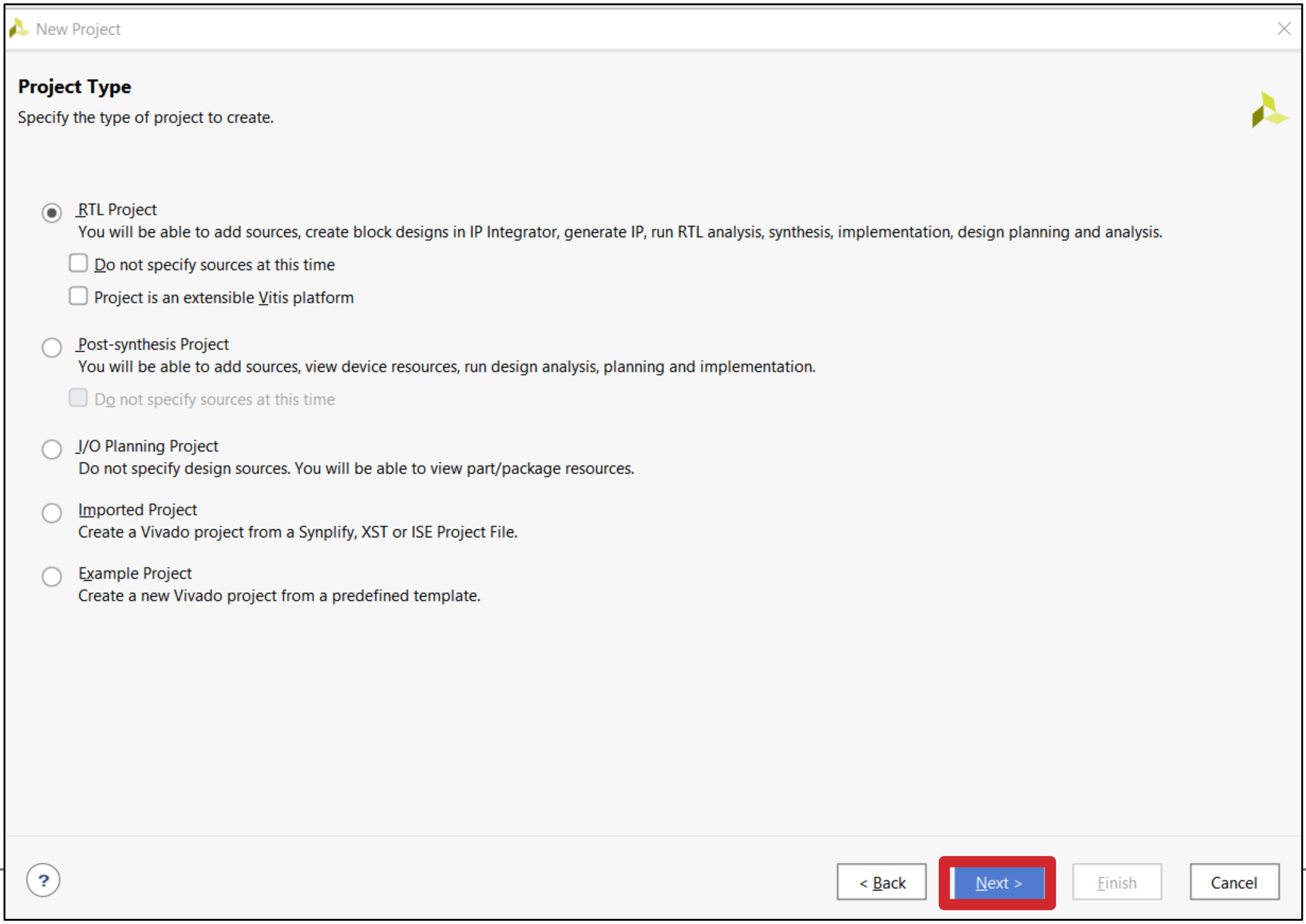
Project will be created at: D:/vivado_projects/CandyVendingMachine

Project name
→ must start with an alphabet
→ can include "_" and numbers

Location
→location of the project
→subfolders can be created

DESIGN EXAMPLE

- Vivado project
  - Creating an new project



New Project

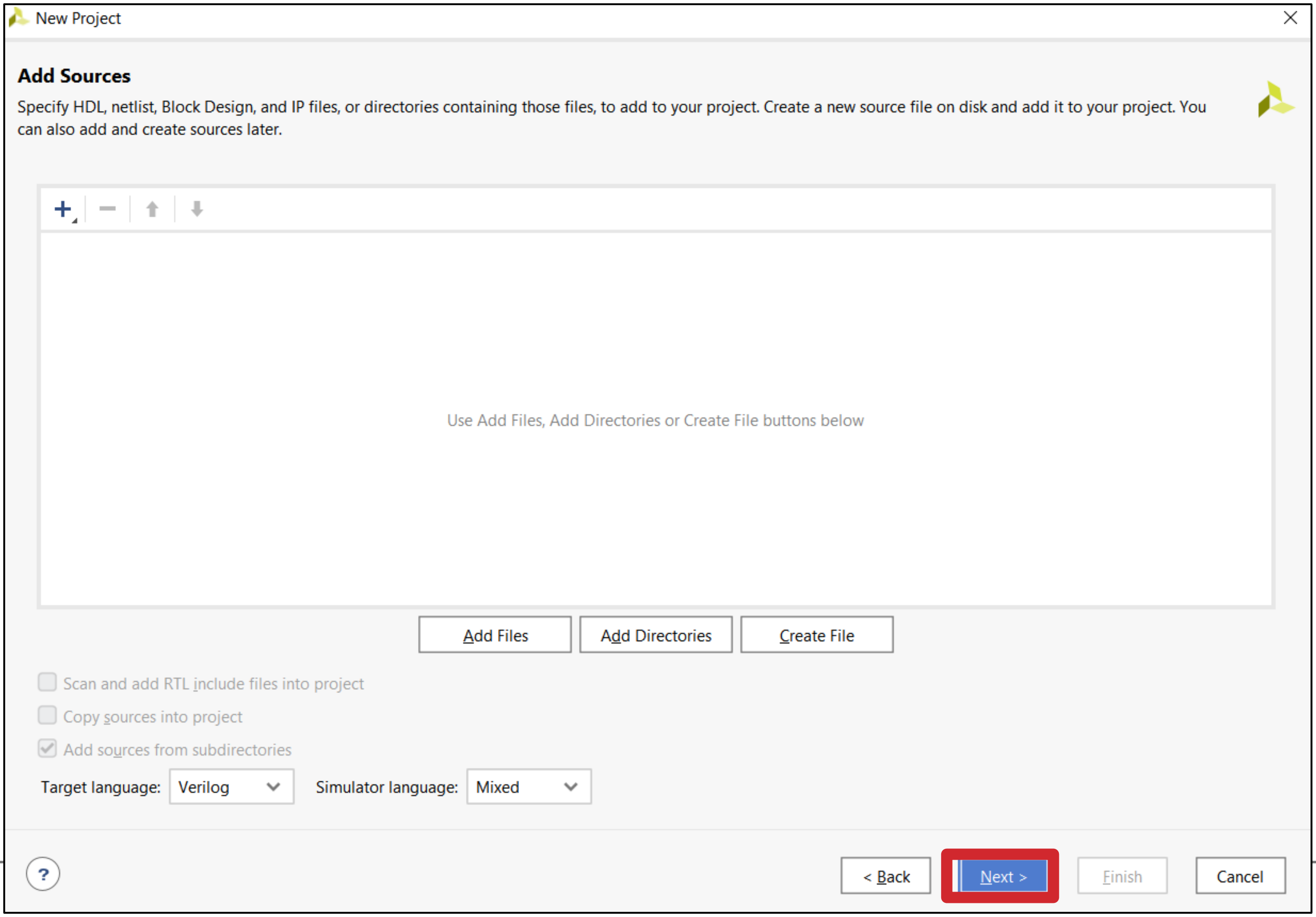
Project Type

Specify the type of project to create.

- [● RTL Project]
  You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
  - ☐ Do not specify sources at this time
  - ☐ Project is an extensible Vitis platform
- ○ Post-synthesis Project
  You will be able to add sources, view device resources, run design analysis, planning and implementation.
  - ☐ Do not specify sources at this time
- ○ I/O Planning Project
  Do not specify design sources. You will be able to view part/package resources.
- ○ Imported Project
  Create a Vivado project from a Synplify, XST or ISE Project File.
- ○ Example Project
  Create a new Vivado project from a predefined template.

DESIGN EXAMPLE

- Vivado project
  - Creating an new project



New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

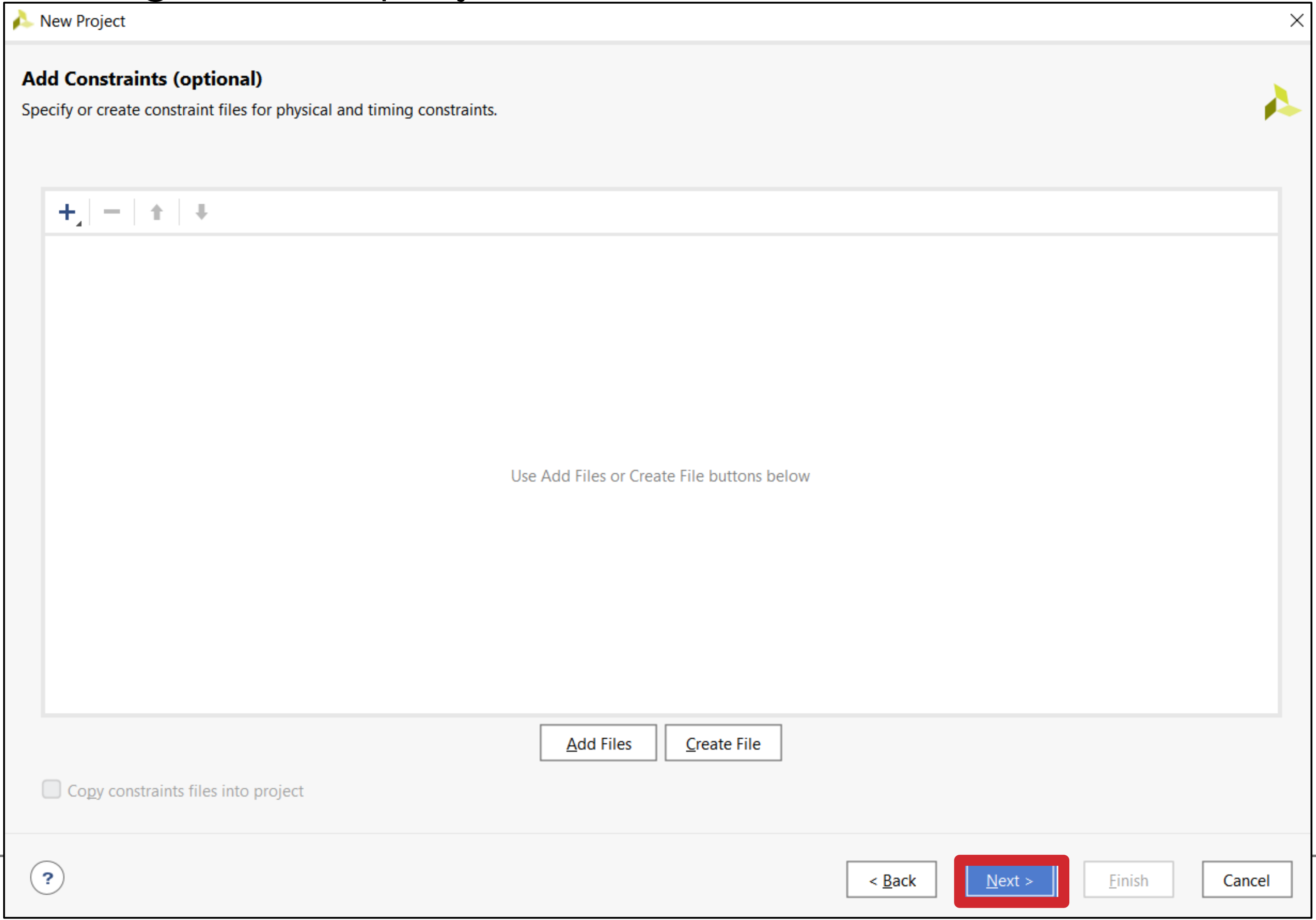
Use Add Files, Add Directories or Create File buttons below

☐ Scan and add RTL include files into project
☐ Copy sources into project
☑ Add sources from subdirectories

Target language: Verilog   Simulator language: Mixed

DESIGN EXAMPLE

- Vivado project
  - Creating an new project



New Project

Add Constraints (optional)

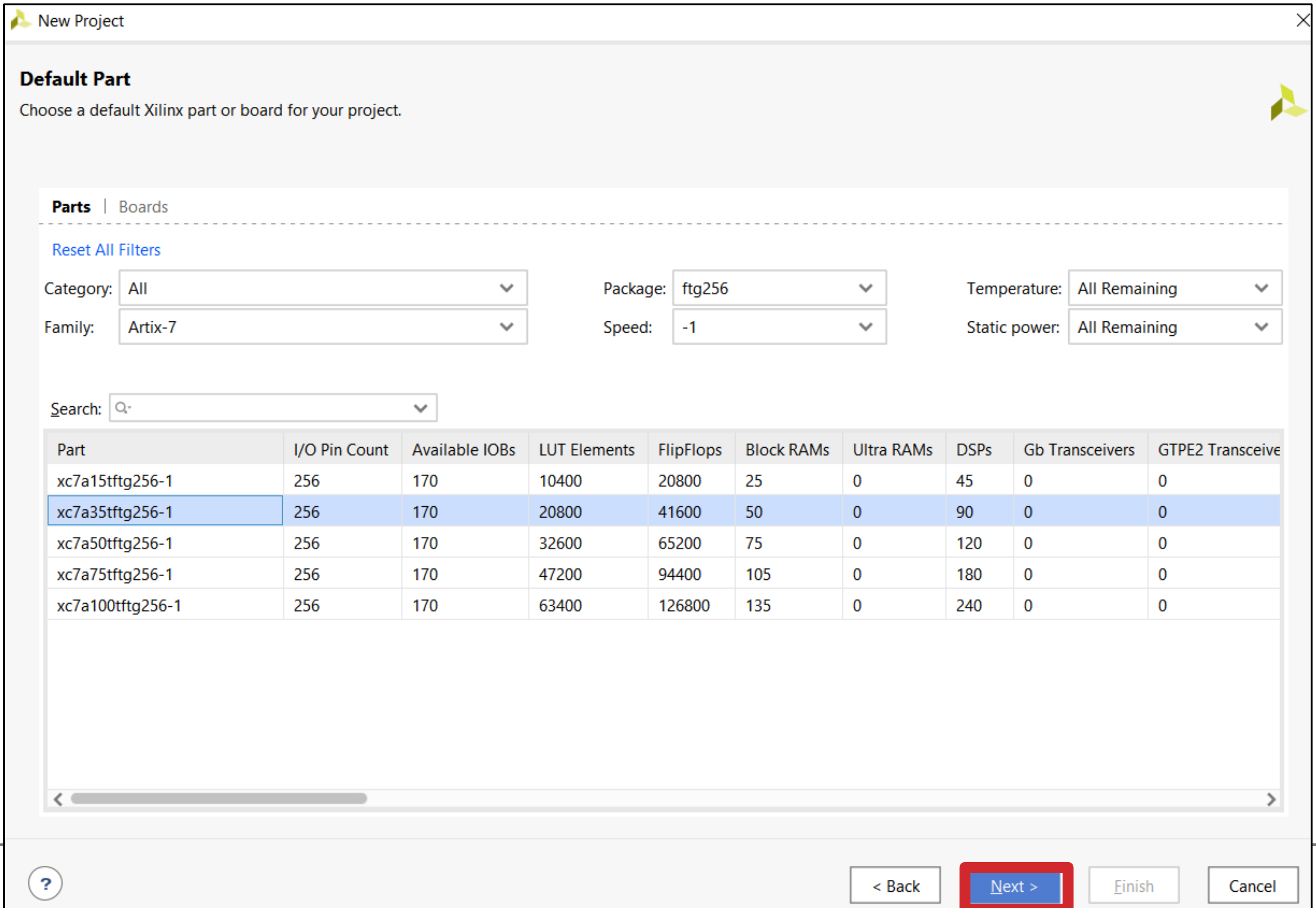
Specify or create constraint files for physical and timing constraints.

Use Add Files or Create File buttons below

DESIGN EXAMPLE

- Vivado project
  - Creating an new project

FPGA Device: Xc7a35tftg256-1



New Project

Default Part

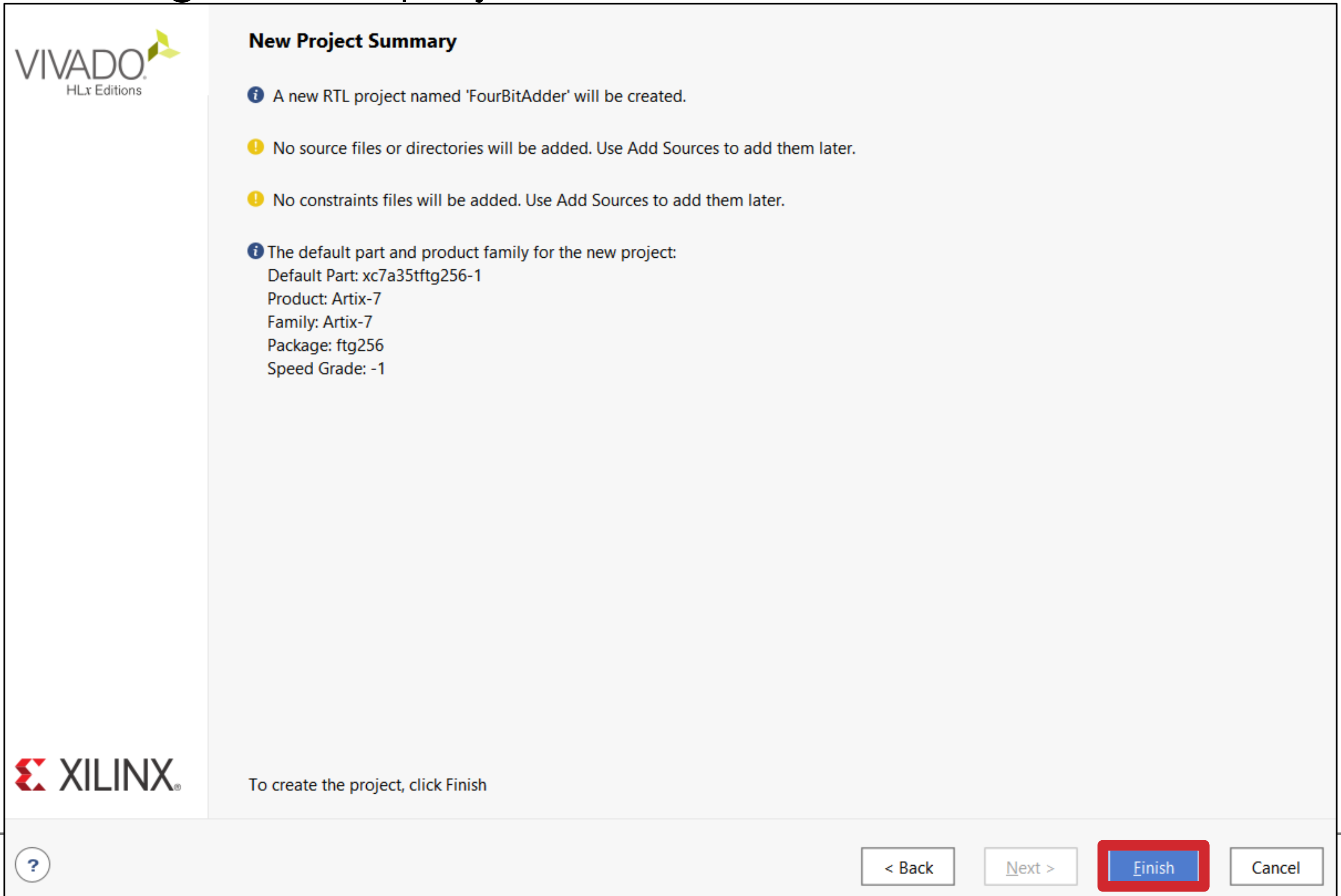
Choose a default Xilinx part or board for your project.

Category: All
Family: Artix-7
Package: ftg256
Speed: -1
Temperature: All Remaining
Static power: All Remaining

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 Transceive
xc7a15tftg256-1	256	170	10400	20800	25	0	45	0	0
xc7a35tftg256-1	256	170	20800	41600	50	0	90	0	0
xc7a50tftg256-1	256	170	32600	65200	75	0	120	0	0
xc7a75tftg256-1	256	170	47200	94400	105	0	180	0	0
xc7a100tftg256-1	256	170	63400	126800	135	0	240	0	0

DESIGN EXAMPLE

- Vivado project
  - Creating an new project



New Project Summary

A new RTL project named 'FourBitAdder' will be created.

No source files or directories will be added. Use Add Sources to add them later.

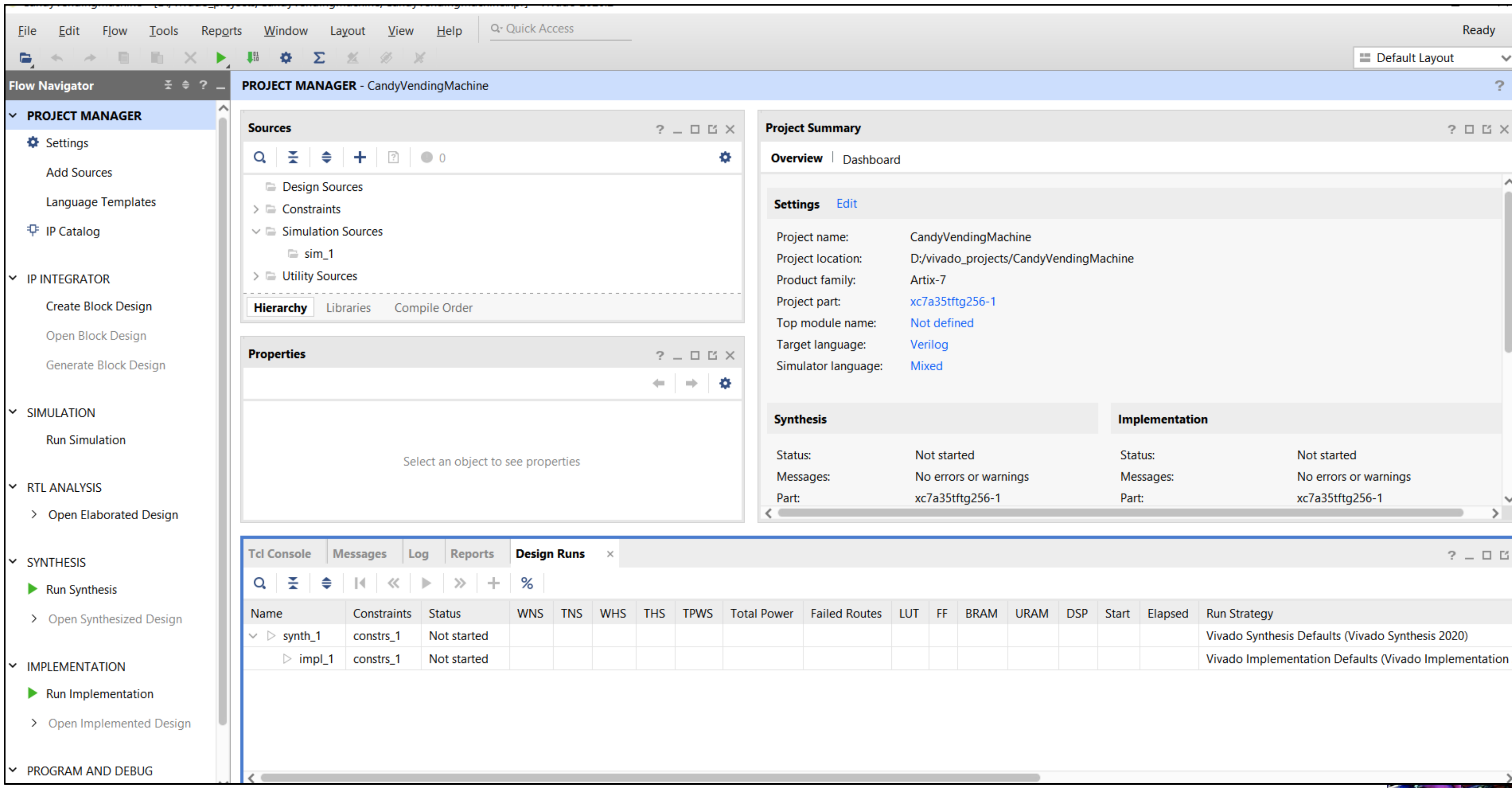
No constraints files will be added. Use Add Sources to add them later.

The default part and product family for the new project:
Default Part: xc7a35tftg256-1
Product: Artix-7
Family: Artix-7
Package: ftg256
Speed Grade: -1

To create the project, click Finish

DESIGN EXAMPLE

- Vivado project



Flow Navigator

- PROJECT MANAGER
  - Settings
  - Add Sources
  - Language Templates
  - IP Catalog
- IP INTEGRATOR
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- SIMULATION
  - Run Simulation
- RTL ANALYSIS
  - Open Elaborated Design
- SYNTHESIS
  - Run Synthesis
  - Open Synthesized Design
- IMPLEMENTATION
  - Run Implementation
  - Open Implemented Design
- PROGRAM AND DEBUG

PROJECT MANAGER - CandyVendingMachine

Sources

- Design Sources
- Constraints
- Simulation Sources
  - sim_1
- Utility Sources

Hierarchy | Libraries | Compile Order

Properties

Select an object to see properties

Project Summary

Overview | Dashboard

Settings Edit

Project name:	CandyVendingMachine
Project location:	D:/vivado_projects/CandyVendingMachine
Product family:	Artix-7
Project part:	xc7a35tftg256-1
Top module name:	Not defined
Target language:	Verilog
Simulator language:	Mixed

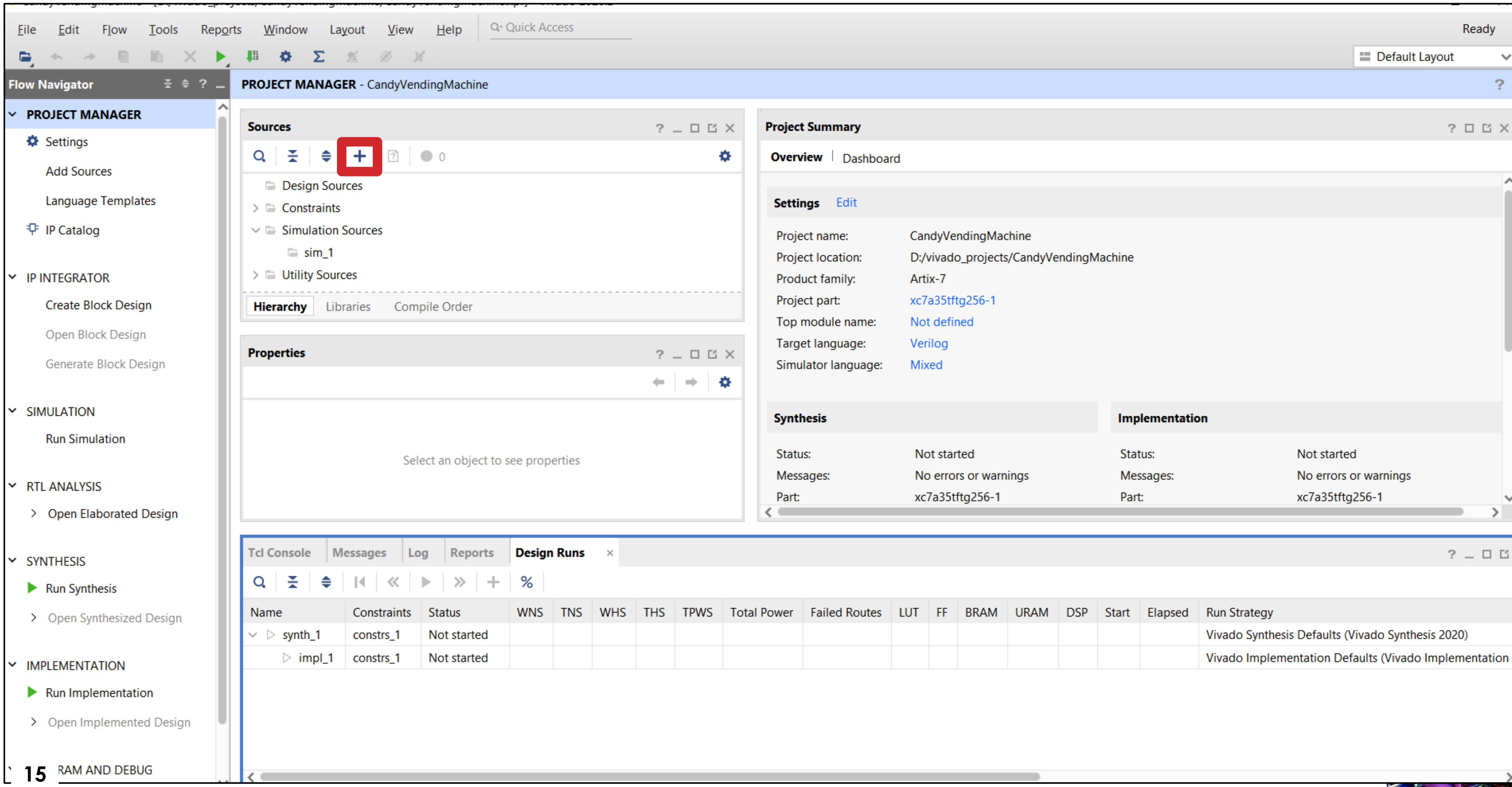
Synthesis		Implementation	
Status:	Not started	Status:	Not started
Messages:	No errors or warnings	Messages:	No errors or warnings
Part:	xc7a35tftg256-1	Part:	xc7a35tftg256-1

Tcl Console | Messages | Log | Reports | **Design Runs**

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2020)
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation

DESIGN EXAMPLE

- Vivado project



File Edit Flow Tools Reports Window Layout View Help Quick Access Ready

Default Layout

Flow Navigator

- PROJECT MANAGER
  - Settings
  - Add Sources
  - Language Templates
  - IP Catalog
- IP INTEGRATOR
  - Create Block Design
  - Open Block Design
  - Generate Block Design
- SIMULATION
  - Run Simulation
- RTL ANALYSIS
  - Open Elaborated Design
- SYNTHESIS
  - Run Synthesis
  - Open Synthesized Design
- IMPLEMENTATION
  - Run Implementation
  - Open Implemented Design
- RAM AND DEBUG

PROJECT MANAGER - CandyVendingMachine

Sources

- Design Sources
- Constraints
- Simulation Sources
  - sim_1
- Utility Sources

Hierarchy | Libraries | Compile Order

Properties

Select an object to see properties

Project Summary

Overview | Dashboard

Settings Edit

Project name:	CandyVendingMachine
Project location:	D:/vivado_projects/CandyVendingMachine
Product family:	Artix-7
Project part:	xc7a35tftg256-1
Top module name:	Not defined
Target language:	Verilog
Simulator language:	Mixed

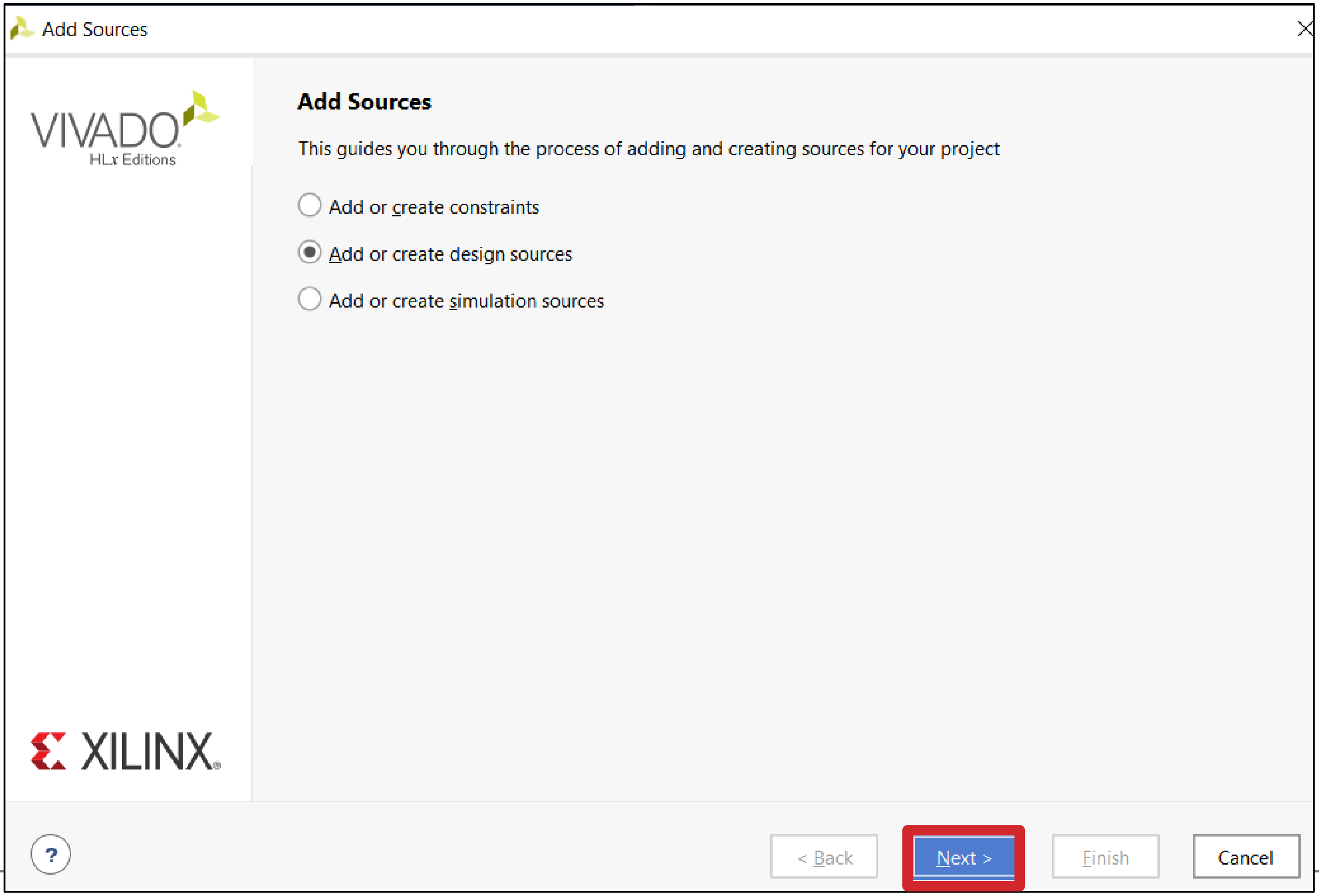
Synthesis		**Implementation**	
Status:	Not started	Status:	Not started
Messages:	No errors or warnings	Messages:	No errors or warnings
Part:	xc7a35tftg256-1	Part:	xc7a35tftg256-1

Tcl Console | Messages | Log | Reports | Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2020)
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation

DESIGN EXAMPLE

- Vivado project



Add Sources

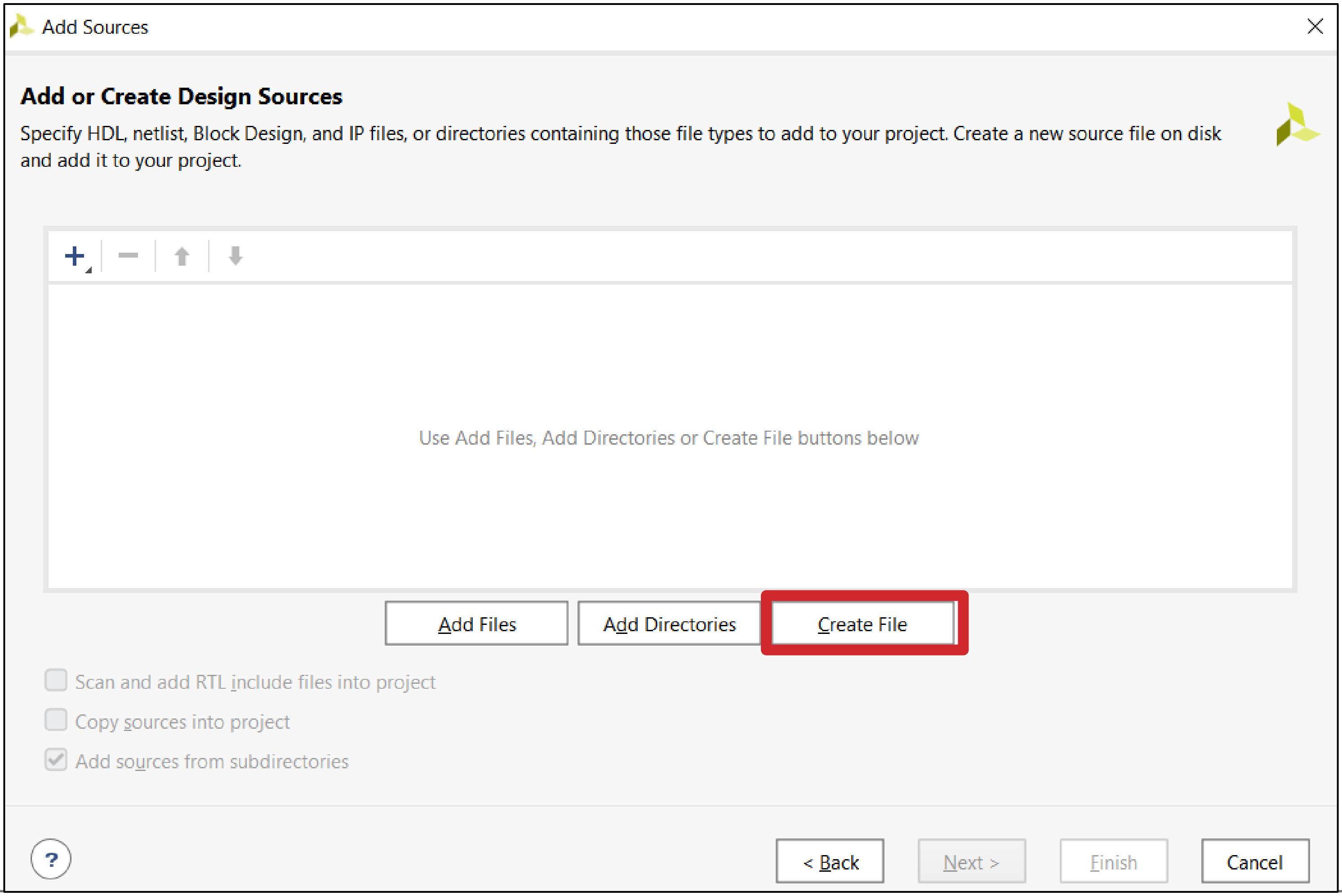
Add Sources

This guides you through the process of adding and creating sources for your project

- ○ Add or create constraints
- ● Add or create design sources
- ○ Add or create simulation sources

DESIGN EXAMPLE

- Vivado project


Add Sources

Add or Create Design Sources

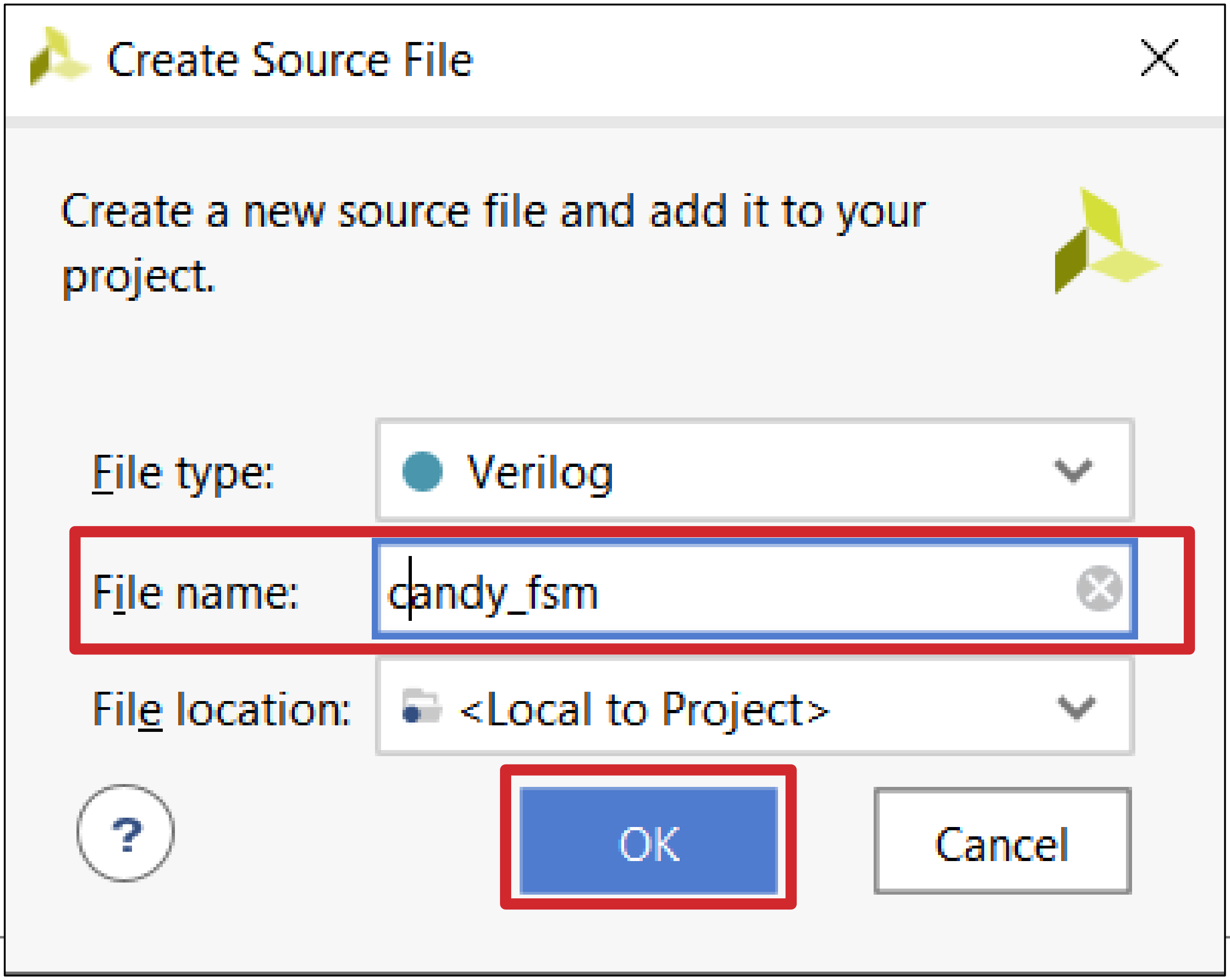
Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

Use Add Files, Add Directories or Create File buttons below

- [ ] Scan and add RTL include files into project
- [ ] Copy sources into project
- [x] Add sources from subdirectories

DESIGN EXAMPLE

❖ Vivado project


Create Source File

Create a new source file and add it to your project.

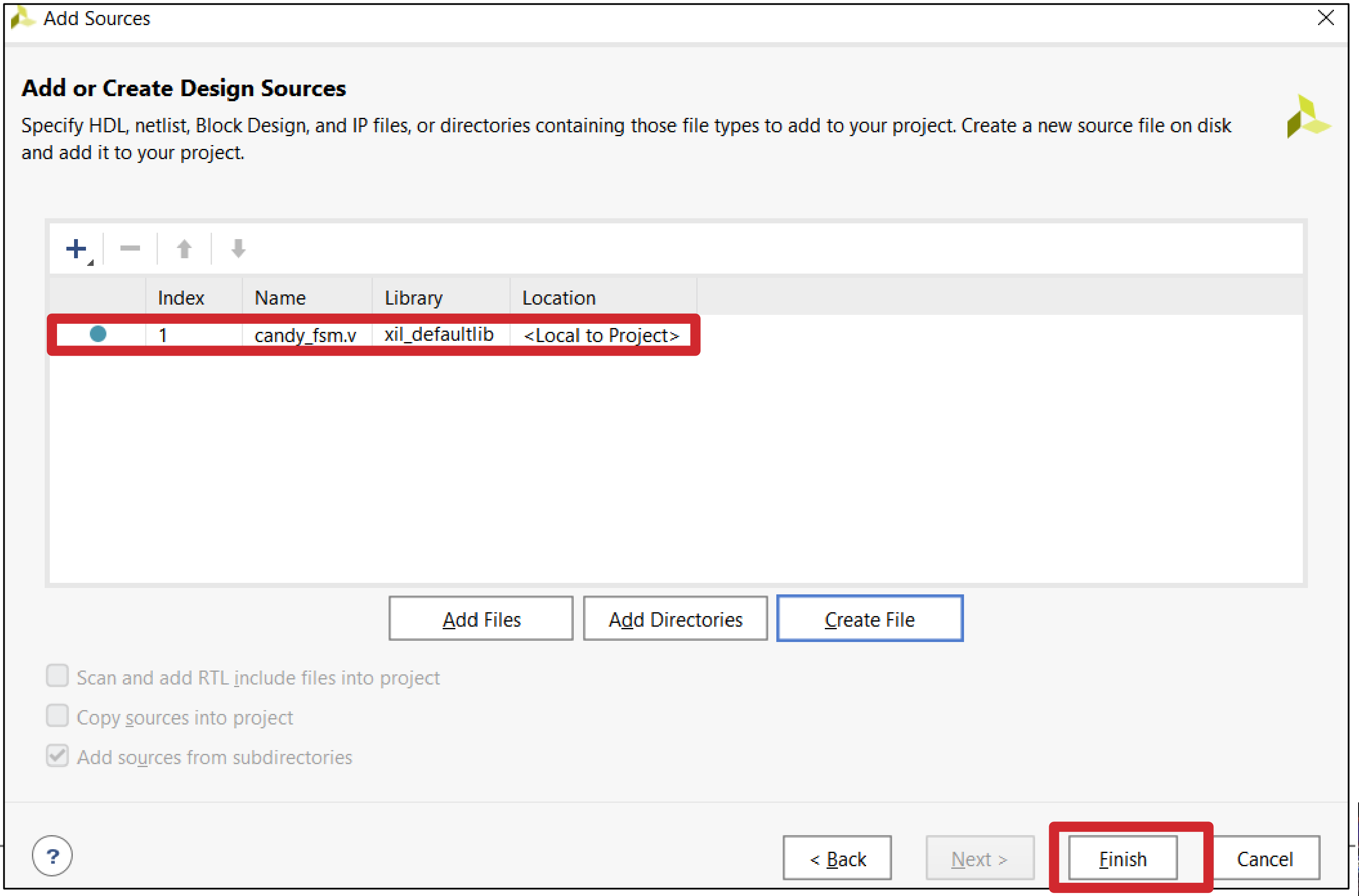
File type: Verilog

File name: candy_fsm

File location: <Local to Project>

DESIGN EXAMPLE

- Vivado project



Add Sources

Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

	Index	Name	Library	Location
●	1	candy_fsm.v	xil_defaultlib	<Local to Project>

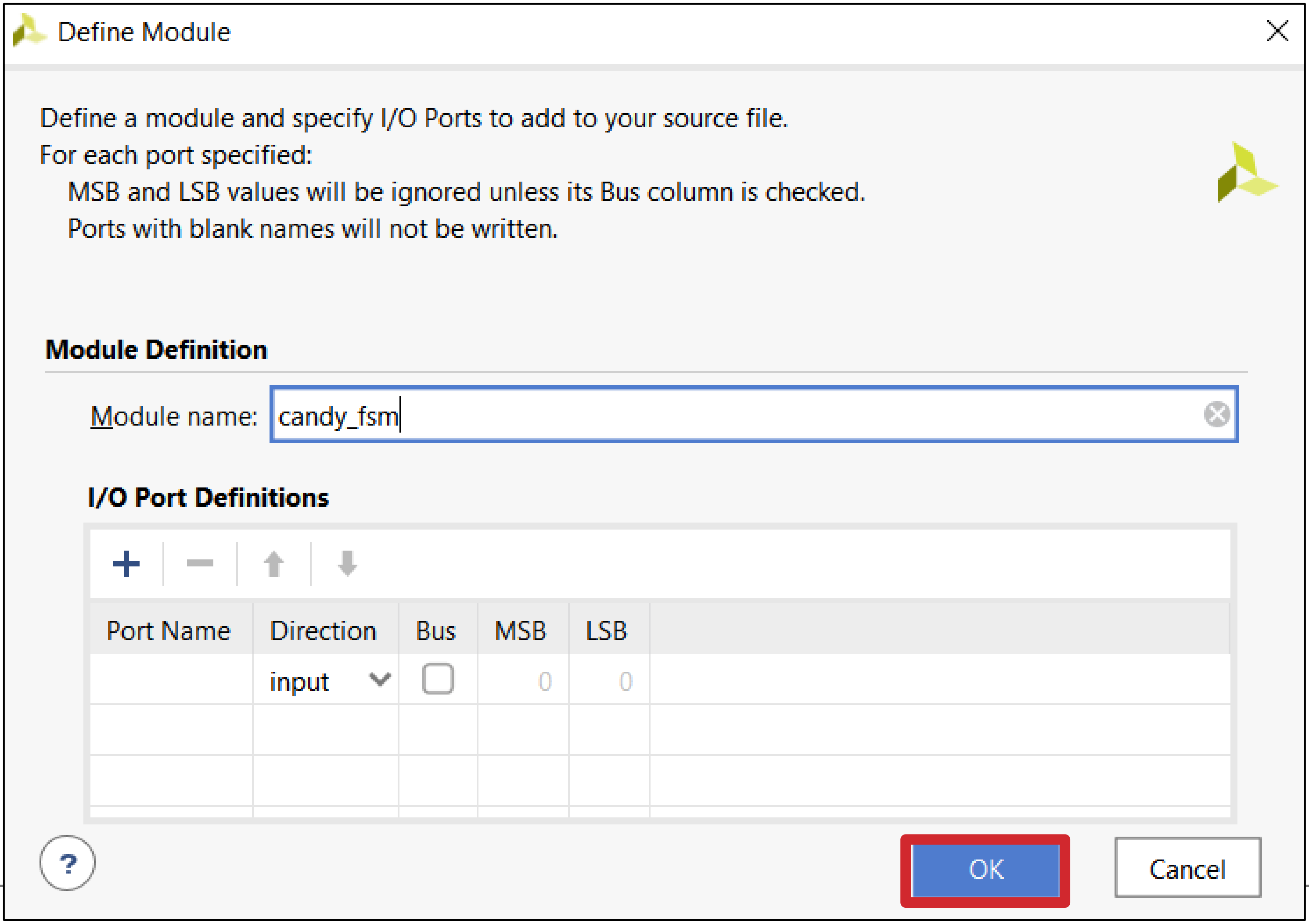
☐ Scan and add RTL include files into project

☐ Copy sources into project

☑ Add sources from subdirectories

DESIGN EXAMPLE

- Vivado project



Define Module

Define a module and specify I/O Ports to add to your source file.
For each port specified:
  MSB and LSB values will be ignored unless its Bus column is checked.
  Ports with blank names will not be written.

Module Definition

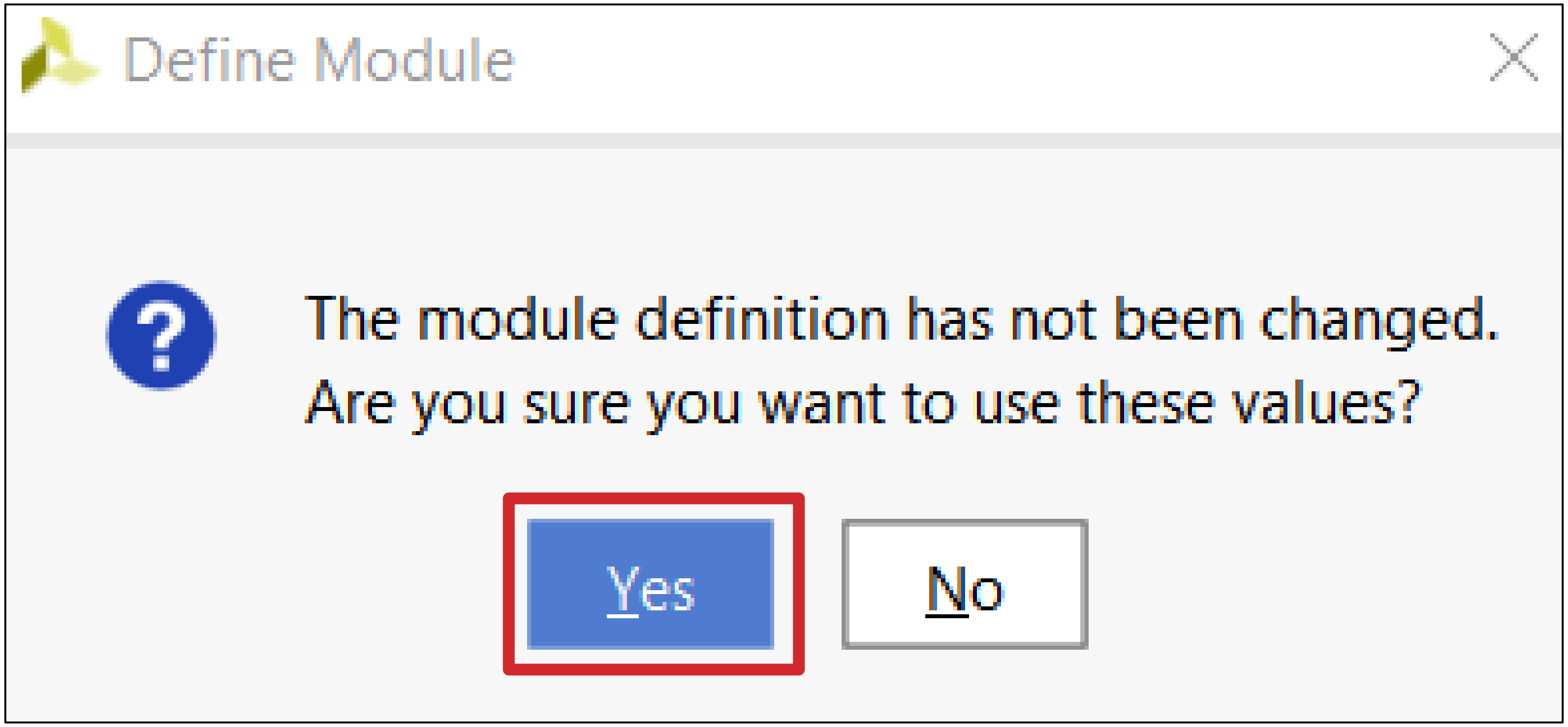
Module name: candy_fsm

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
	input	☐	0	0

DESIGN EXAMPLE

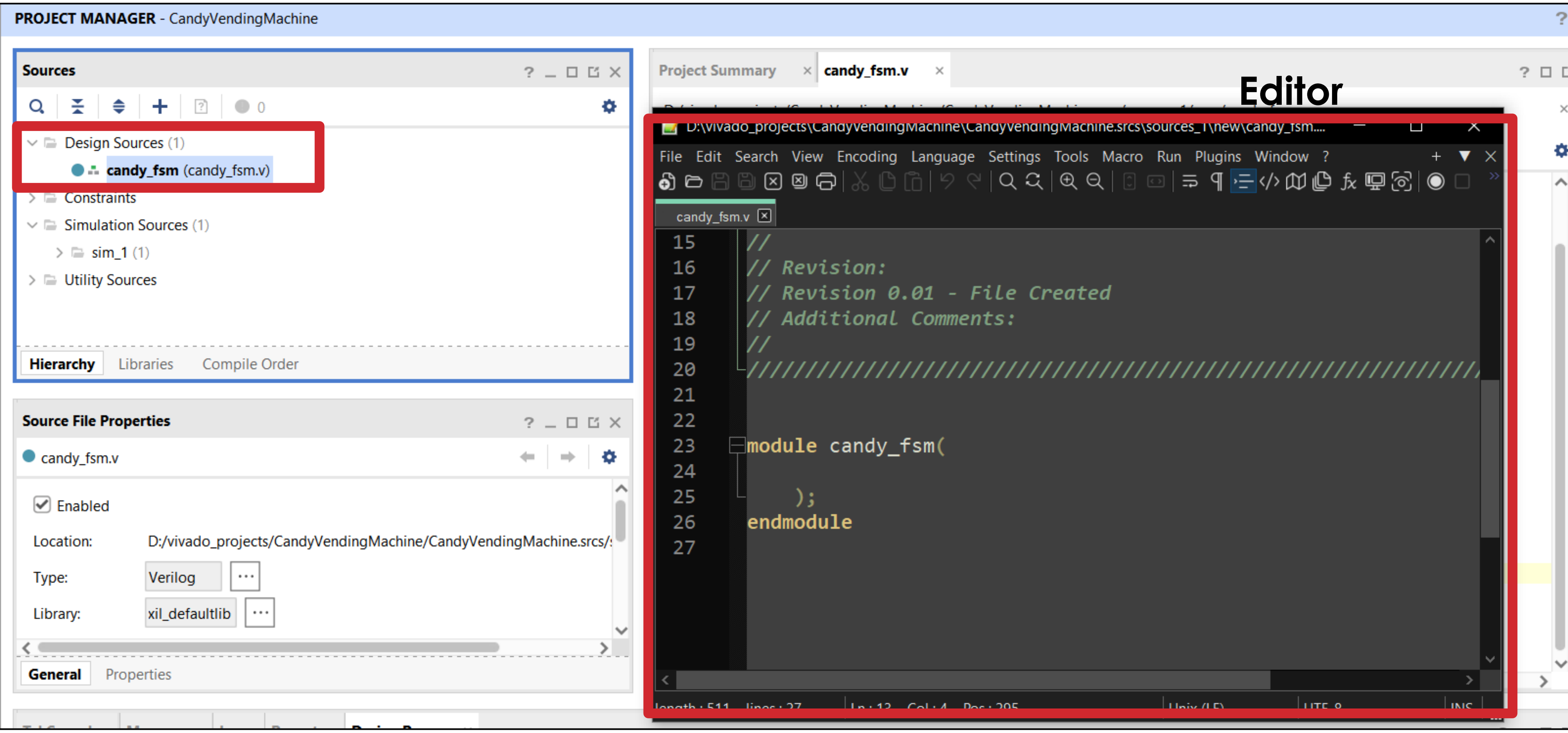
- Vivado project


Define Module

The module definition has not been changed.
Are you sure you want to use these values?

DESIGN EXAMPLE

- Vivado project



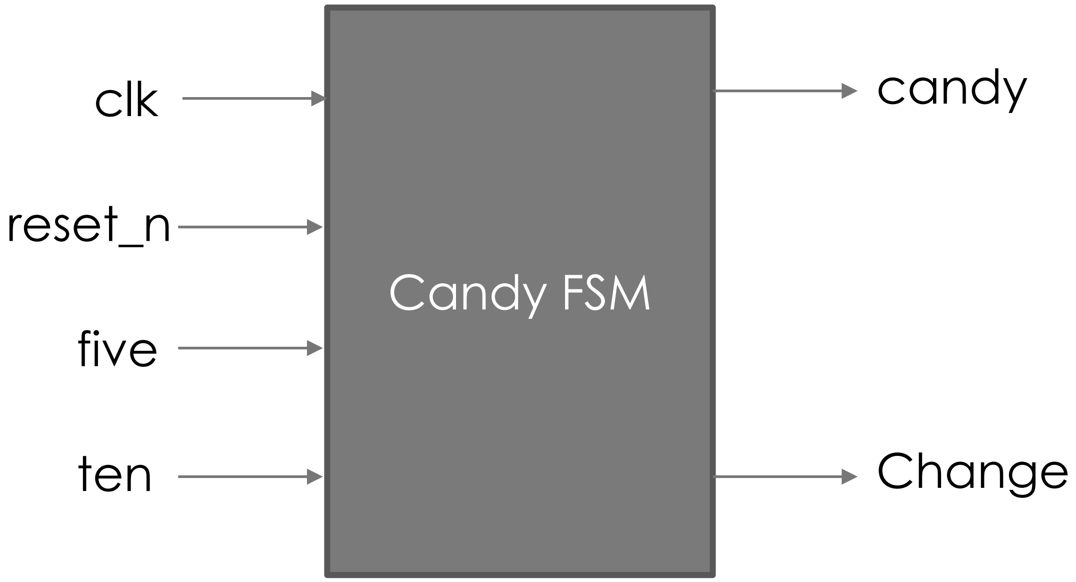
Enabled
Location: D:/vivado_projects/CandyVendingMachine/CandyVendingMachine.srcs/
Type: Verilog
Library: xil_defaultlib

D:\vivado_projects\CandyVendingMachine\CandyVendingMachine.srcs\sources_1\new\candy_fsm....

```
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////


module candy_fsm(

    );
endmodule
```

DESIGN EXAMPLE

- Candy Vending Machine FSM Block Diagram
  - Block Diagram and Signal Description

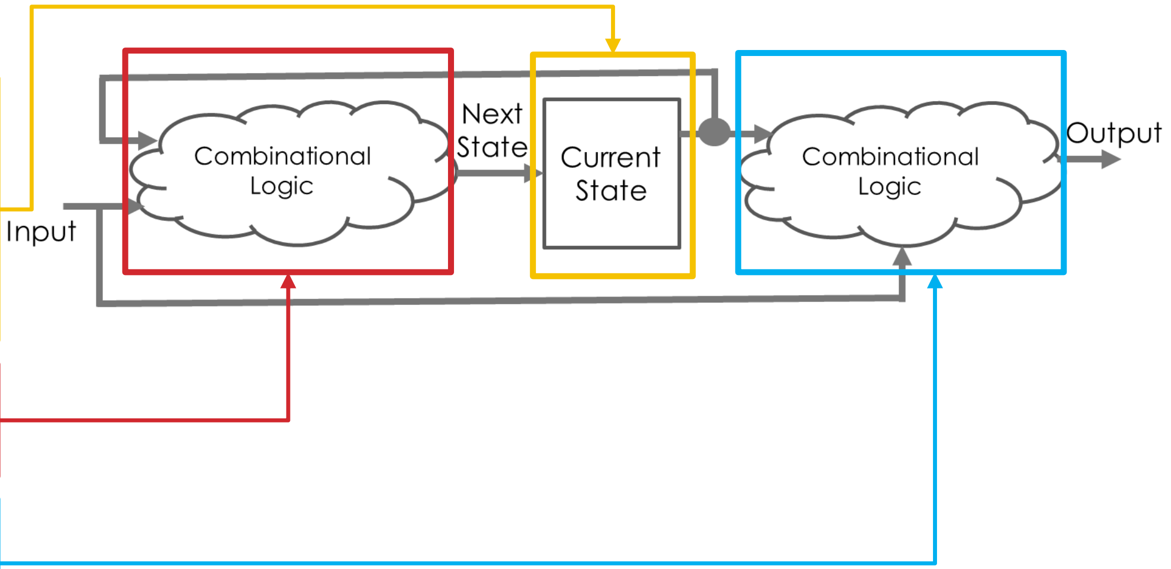


Signal	Description
clk	Input Clock
reset_n	Input negedge edge reset
five	Input Five coin
ten	Input Ten coin
candy	Output candy
change	Output change

DESIGN EXAMPLE

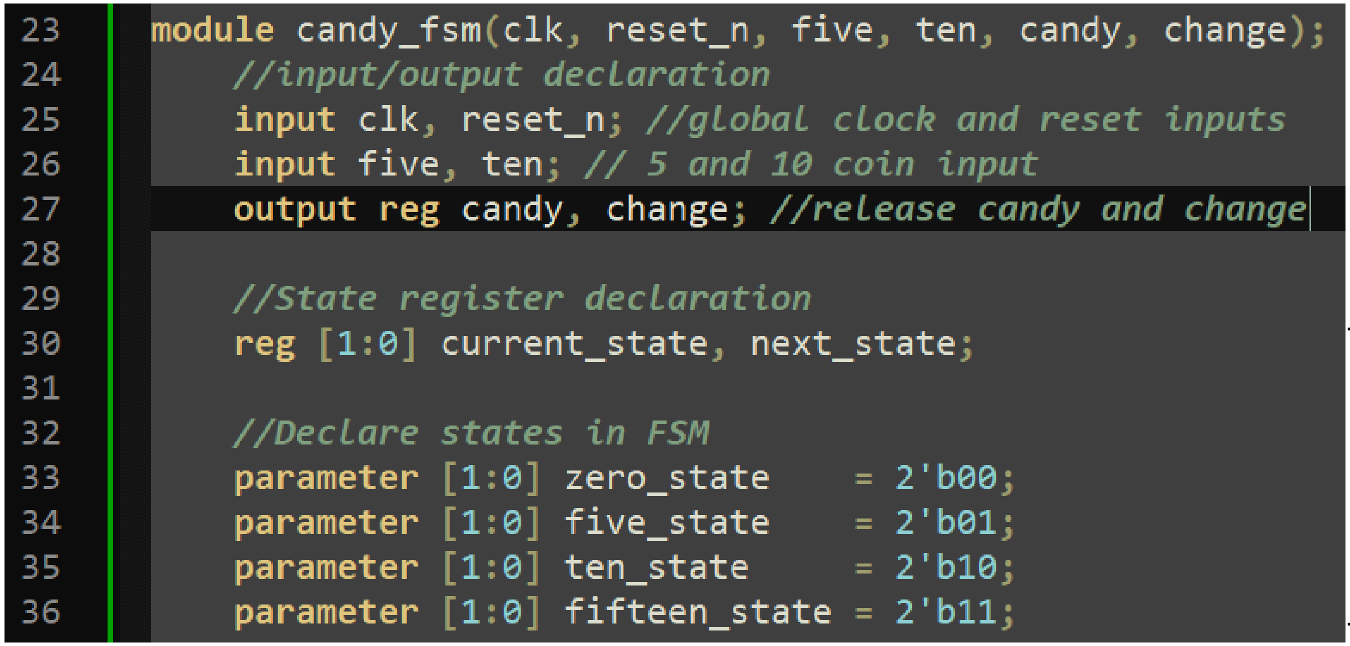
- Candy Vending Machine FSM
  - Mealy Coding Style
    - Using 1 sequential process and 2 combinatorial processes

```
module mudule_name (//input/outputs);
    //input/output declaration
    input ...;
    output ...;

    //state register declaration
    reg [...] current_state, next_state;

    //Part 1: Sequential Process
    always@(posedge clk or negedge reset_n) begin
        if(!reset_n) begin
            //<Initialize current state>
        end
        else begin
            //<Update current state>
        end
    end

    //Part 2: Combinational Logic
    always@(current_state or inputs) begin
        //<compute next state function>
    end

    //Part 3: Combinational Logic
    always@(current_state or inputs) begin
        //<compute output function>
    end

endmodule
```



DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Verilog Code for Candy Vending Machine FSM
    - I/O and state registers declaration



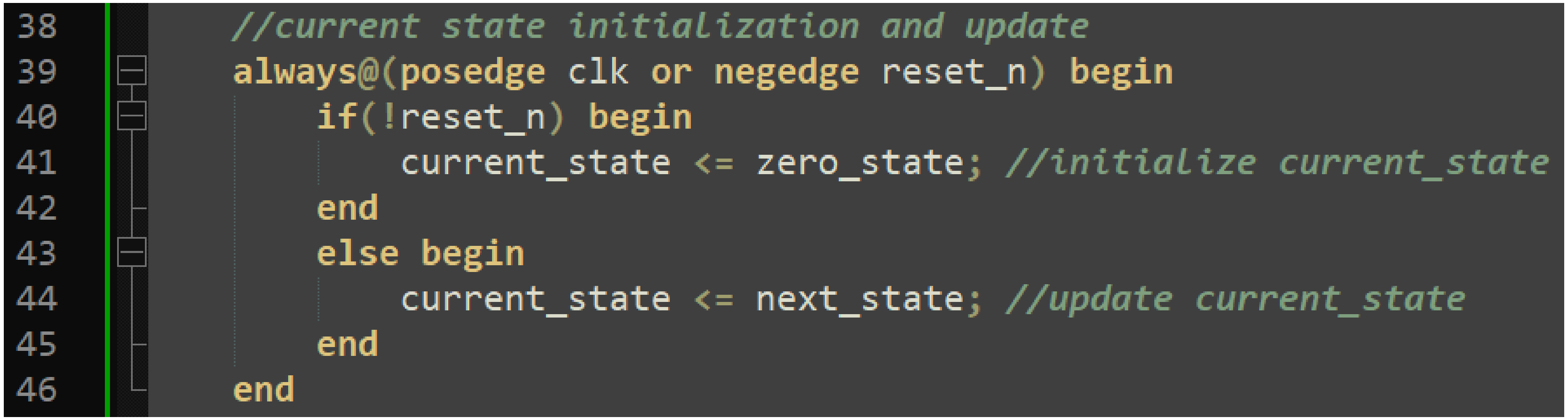
```verilog
module candy_fsm(clk, reset_n, five, ten, candy, change);
    //input/output declaration
    input clk, reset_n; //global clock and reset inputs
    input five, ten; // 5 and 10 coin input
    output reg candy, change; //release candy and change

    //State register declaration
    reg [1:0] current_state, next_state;

    //Declare states in FSM
    parameter [1:0] zero_state    = 2'b00;
    parameter [1:0] five_state    = 2'b01;
    parameter [1:0] ten_state     = 2'b10;
    parameter [1:0] fifteen_state = 2'b11;
```

Global signals and Input/output declaration

State registers and state declaration

DESIGN EXAMPLE

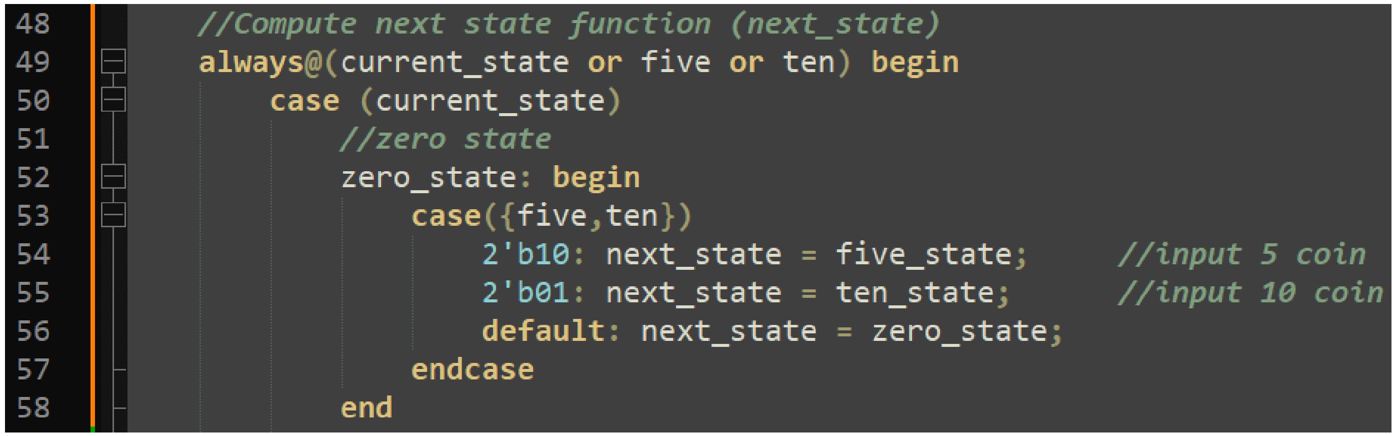
- Candy Vending Machine Project Project
  - Verilog Code for Candy Vending Machine FSM
    - Current state register initialization and updating



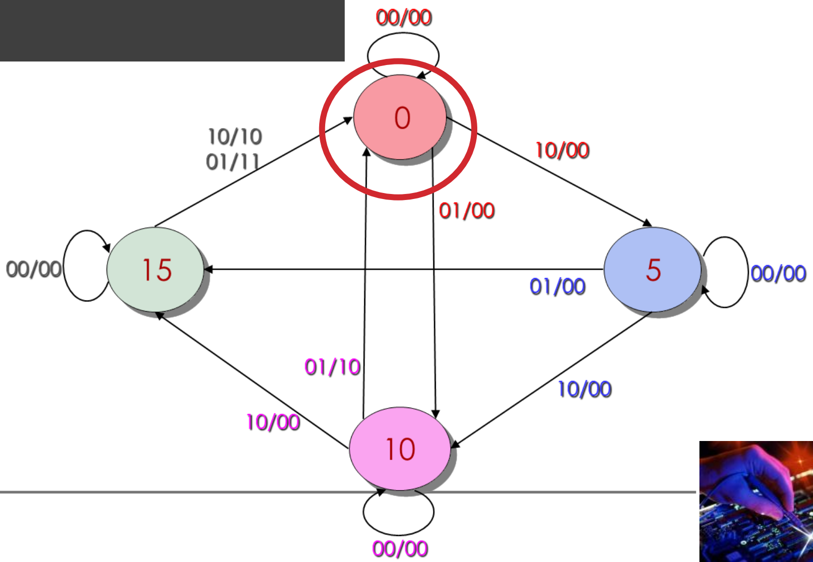
```
    //current state initialization and update
    always@(posedge clk or negedge reset_n) begin
        if(!reset_n) begin
            current_state <= zero_state; //initialize current_state
        end
        else begin
            current_state <= next_state; //update current_state
        end
    end
```

DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Verilog Code for Candy Vending Machine FSM
    - Next state function computation

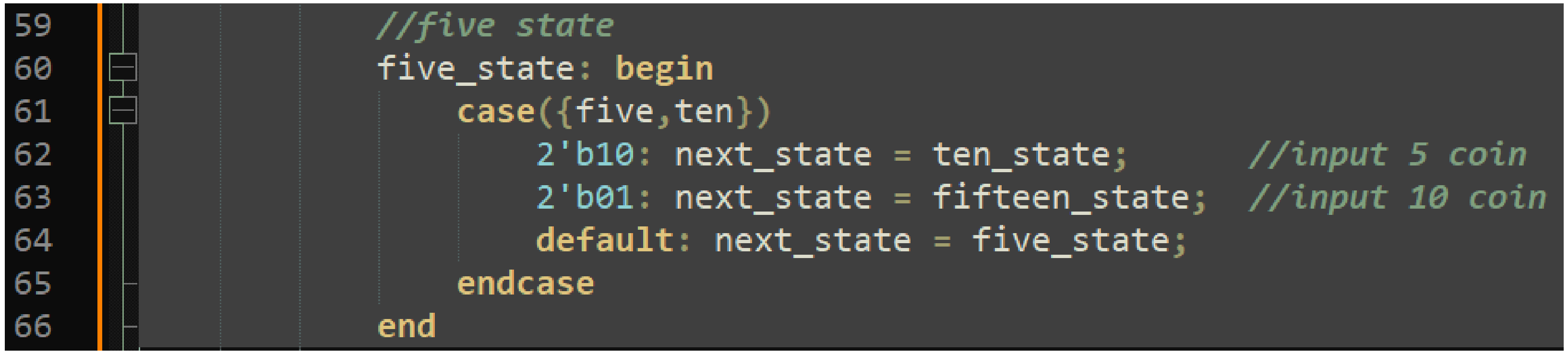


```
    //Compute next state function (next_state)
    always@(current_state or five or ten) begin
        case (current_state)
            //zero state
            zero_state: begin
                case({five,ten})
                    2'b10: next_state = five_state;     //input 5 coin
                    2'b01: next_state = ten_state;      //input 10 coin
                    default: next_state = zero_state;
                endcase
            end
```

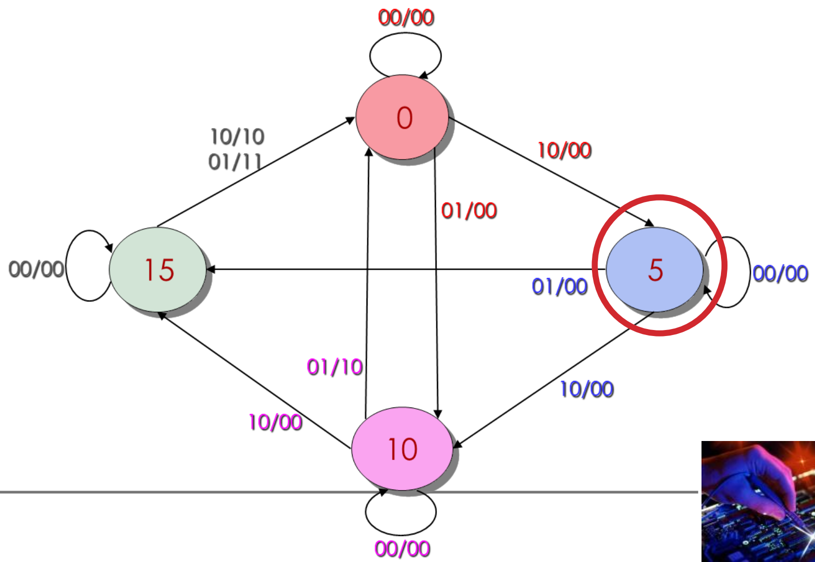


DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Verilog Code for Candy Vending Machine FSM
    - Next state function computation

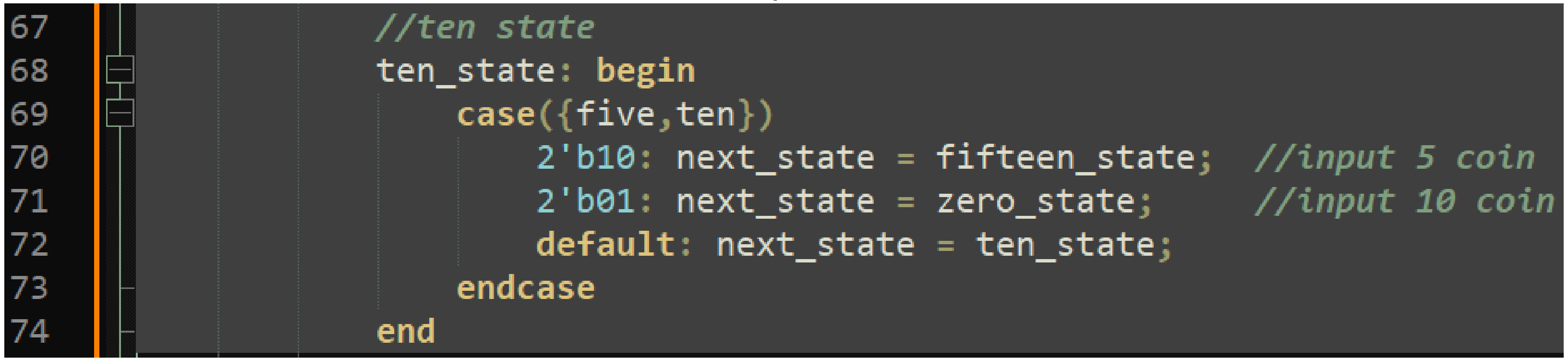


```
        //five state
        five_state: begin
            case({five,ten})
                2'b10: next_state = ten_state;       //input 5 coin
                2'b01: next_state = fifteen_state;   //input 10 coin
                default: next_state = five_state;
            endcase
        end
```

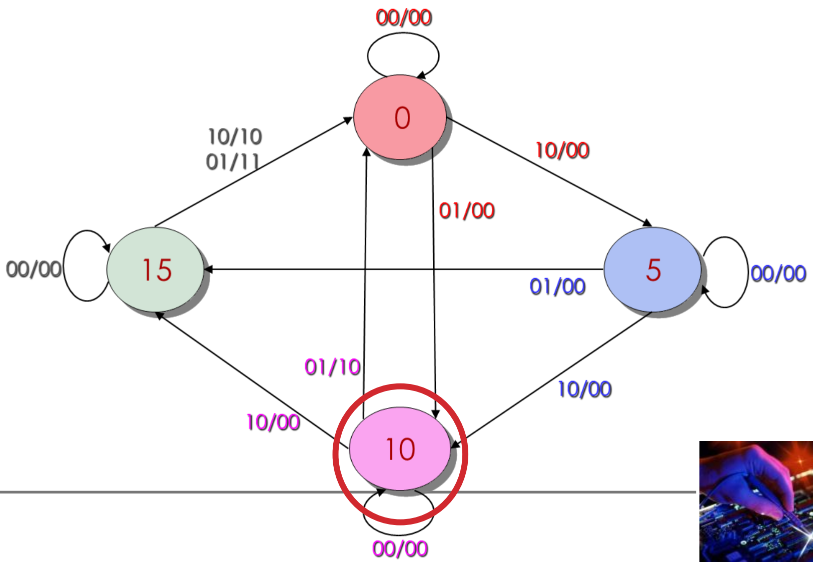


DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Verilog Code for Candy Vending Machine FSM
    - Next state function computation

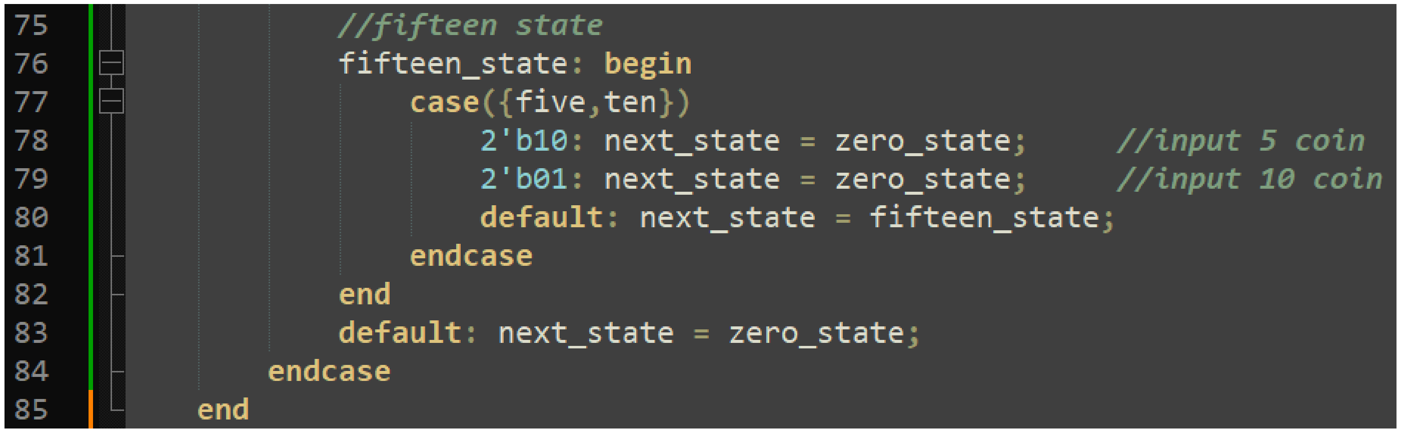


```
        //ten state
        ten_state: begin
            case({five,ten})
                2'b10: next_state = fifteen_state;  //input 5 coin
                2'b01: next_state = zero_state;     //input 10 coin
                default: next_state = ten_state;
            endcase
        end
```

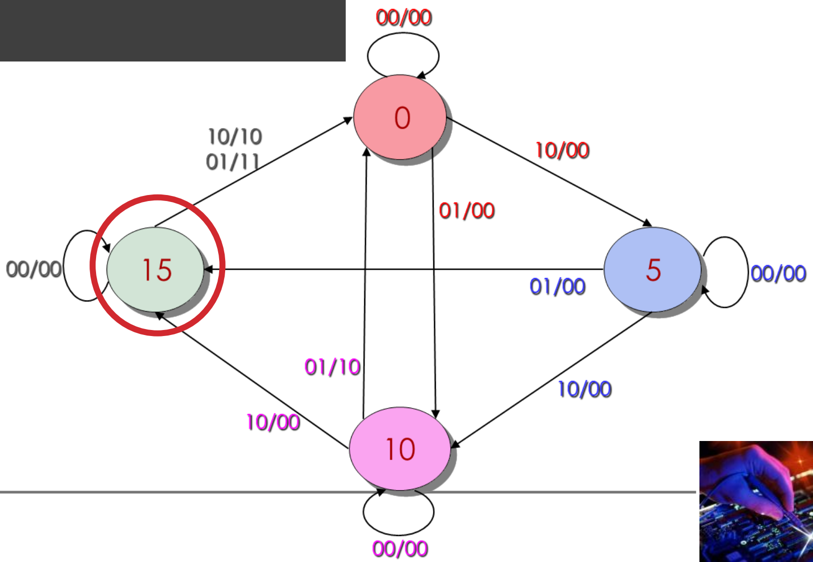


DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Verilog Code for Candy Vending Machine FSM
    - Next state function computation

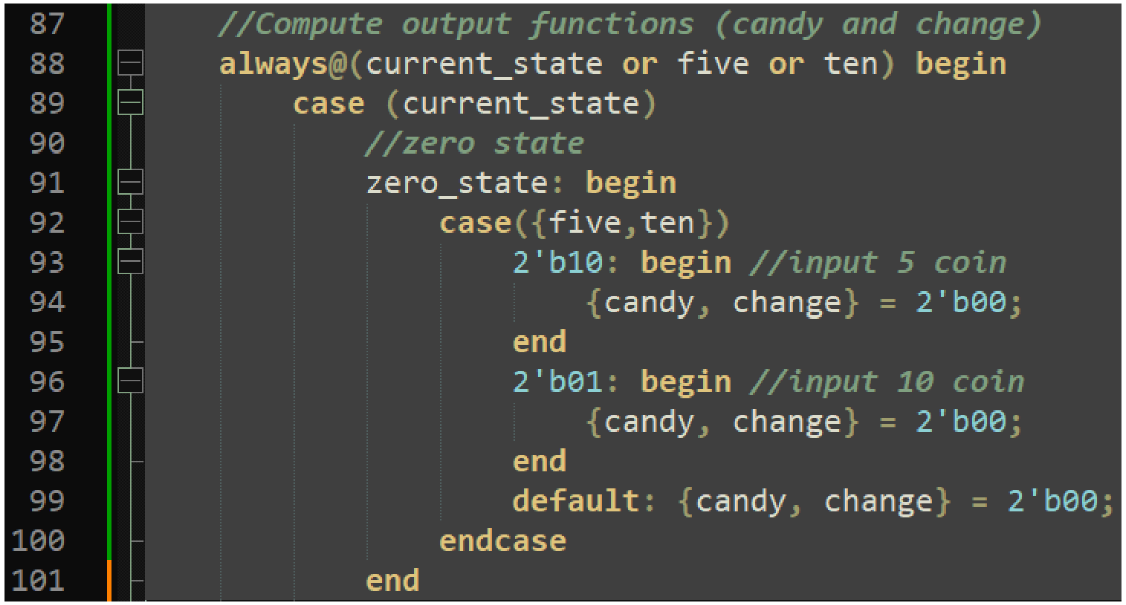


```
        //fifteen state
        fifteen_state: begin
            case({five,ten})
                2'b10: next_state = zero_state;     //input 5 coin
                2'b01: next_state = zero_state;     //input 10 coin
                default: next_state = fifteen_state;
            endcase
        end
        default: next_state = zero_state;
    endcase
end
```

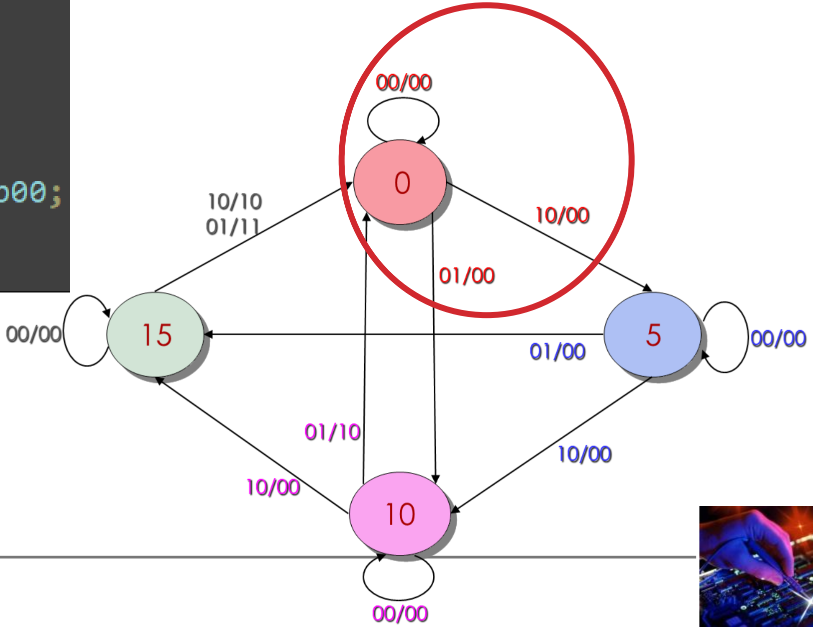


DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Verilog Code for Candy Vending Machine FSM
    - Output function computation

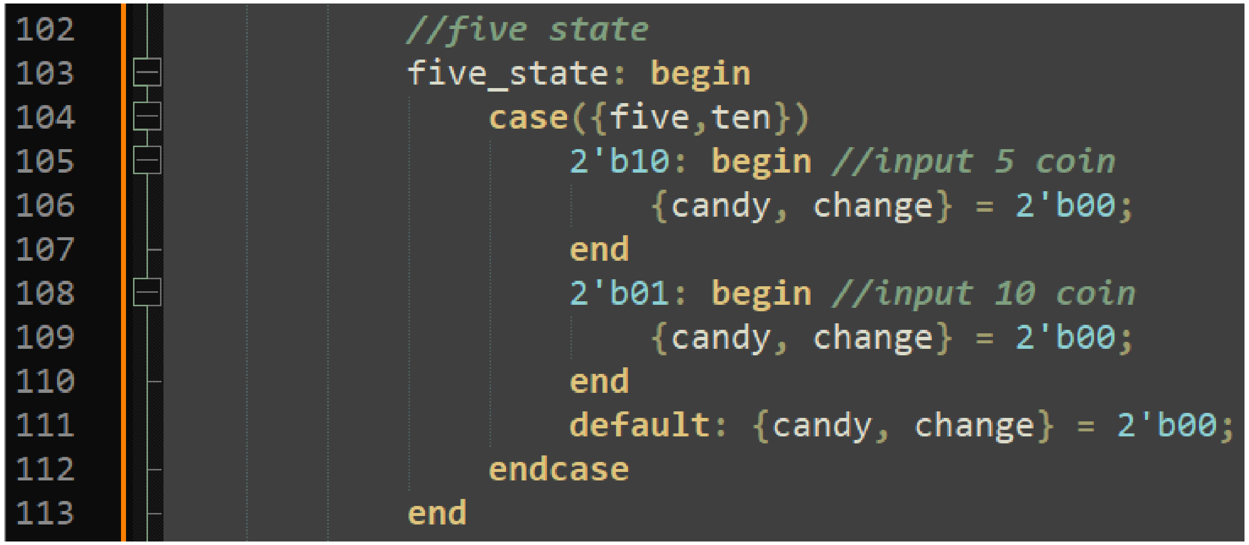


```
//Compute output functions (candy and change)
always@(current_state or five or ten) begin
    case (current_state)
        //zero state
        zero_state: begin
            case({five,ten})
                2'b10: begin //input 5 coin
                    {candy, change} = 2'b00;
                end
                2'b01: begin //input 10 coin
                    {candy, change} = 2'b00;
                end
                default: {candy, change} = 2'b00;
            endcase
        end
```

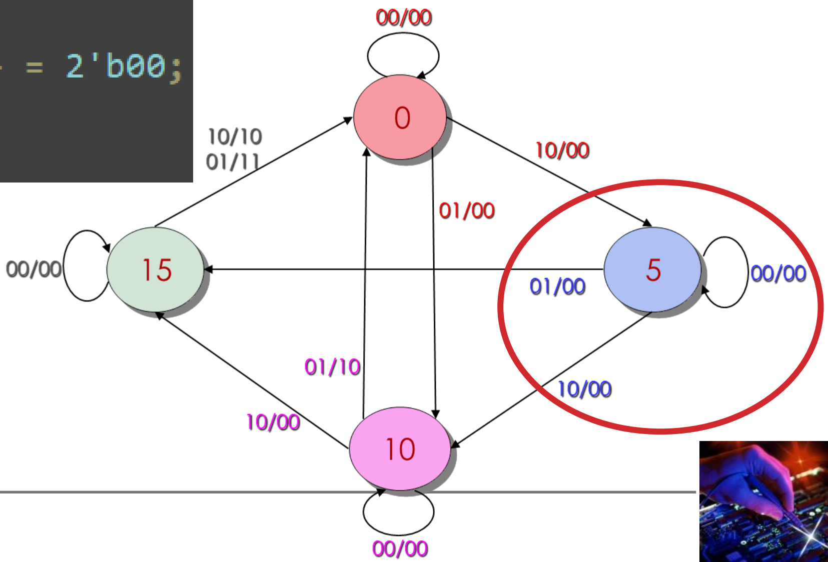


DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Verilog Code for Candy Vending Machine FSM
    - Output function computation

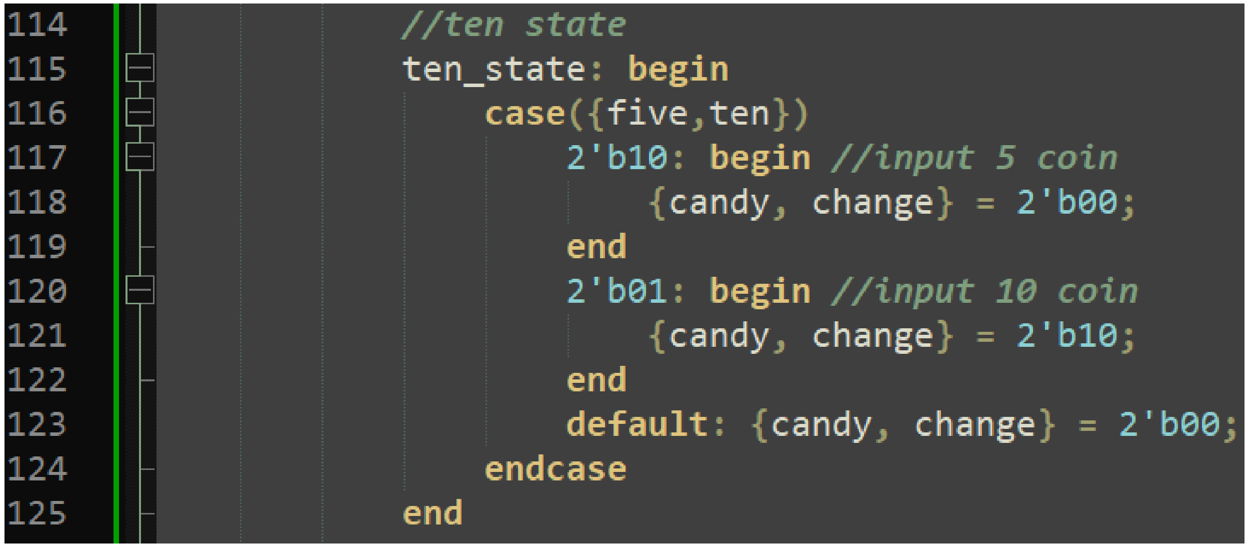


```verilog
            //five state
            five_state: begin
                case({five,ten})
                    2'b10: begin //input 5 coin
                        {candy, change} = 2'b00;
                    end
                    2'b01: begin //input 10 coin
                        {candy, change} = 2'b00;
                    end
                    default: {candy, change} = 2'b00;
                endcase
            end
```

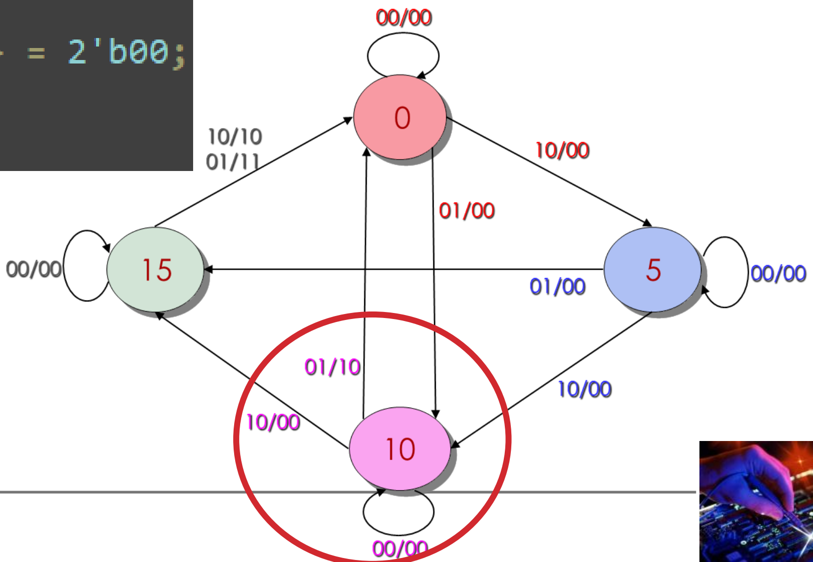


DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Verilog Code for Candy Vending Machine FSM
    - Output function computation

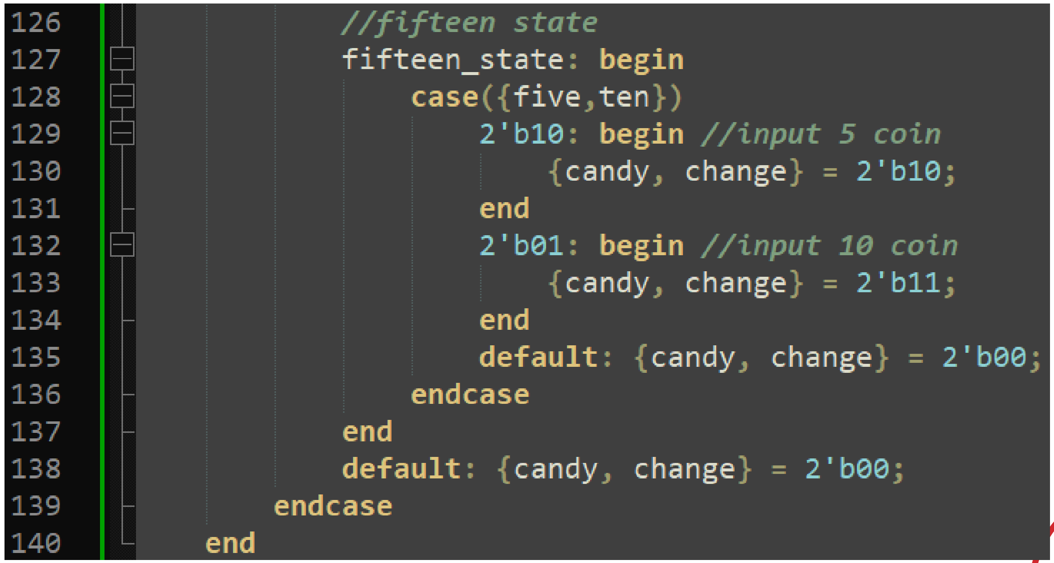


```
            //ten state
            ten_state: begin
                case({five,ten})
                    2'b10: begin //input 5 coin
                        {candy, change} = 2'b00;
                    end
                    2'b01: begin //input 10 coin
                        {candy, change} = 2'b10;
                    end
                    default: {candy, change} = 2'b00;
                endcase
            end
```

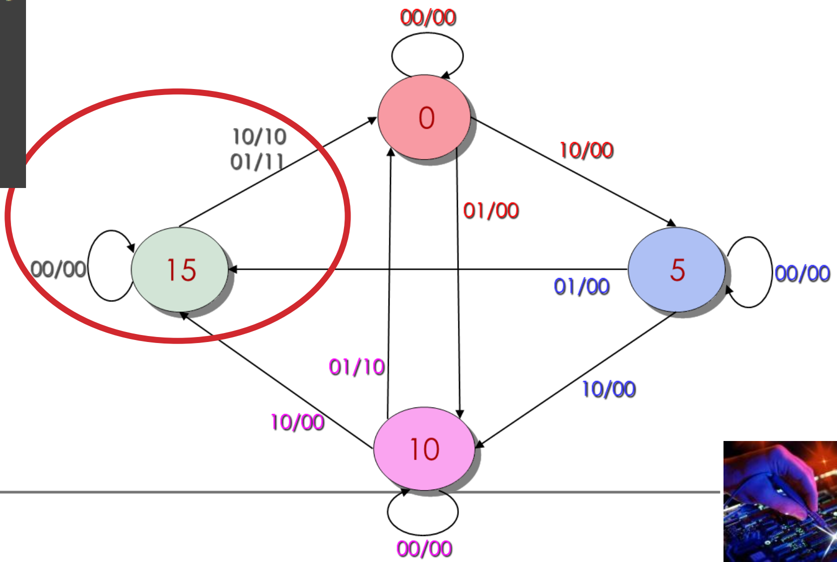


DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Verilog Code for Candy Vending Machine FSM
    - Output function computation

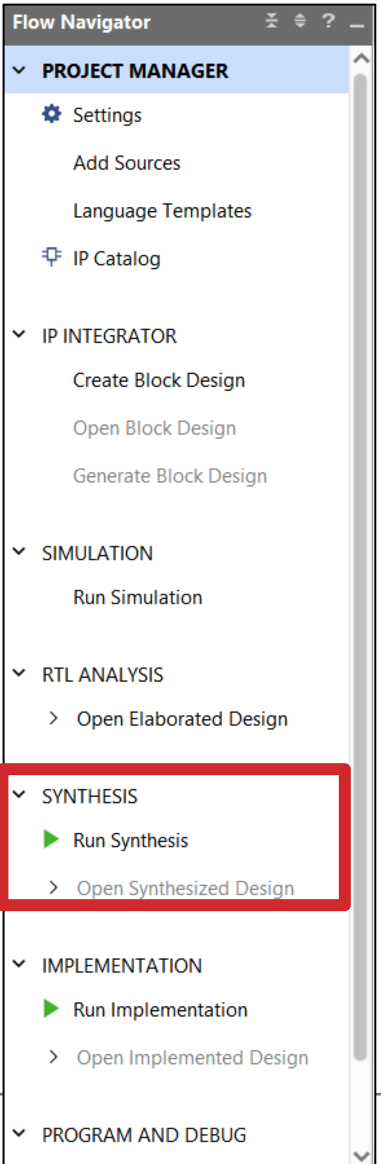


```
                //fifteen state
                fifteen_state: begin
                    case({five,ten})
                        2'b10: begin //input 5 coin
                            {candy, change} = 2'b10;
                        end
                        2'b01: begin //input 10 coin
                            {candy, change} = 2'b11;
                        end
                        default: {candy, change} = 2'b00;
                    endcase
                end
                default: {candy, change} = 2'b00;
            endcase
        end
```



DESIGN EXAMPLE

- Vivado project
  - Logic synthesis


Flow Navigator

PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog

IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design

SIMULATION
Run Simulation

RTL ANALYSIS
Open Elaborated Design

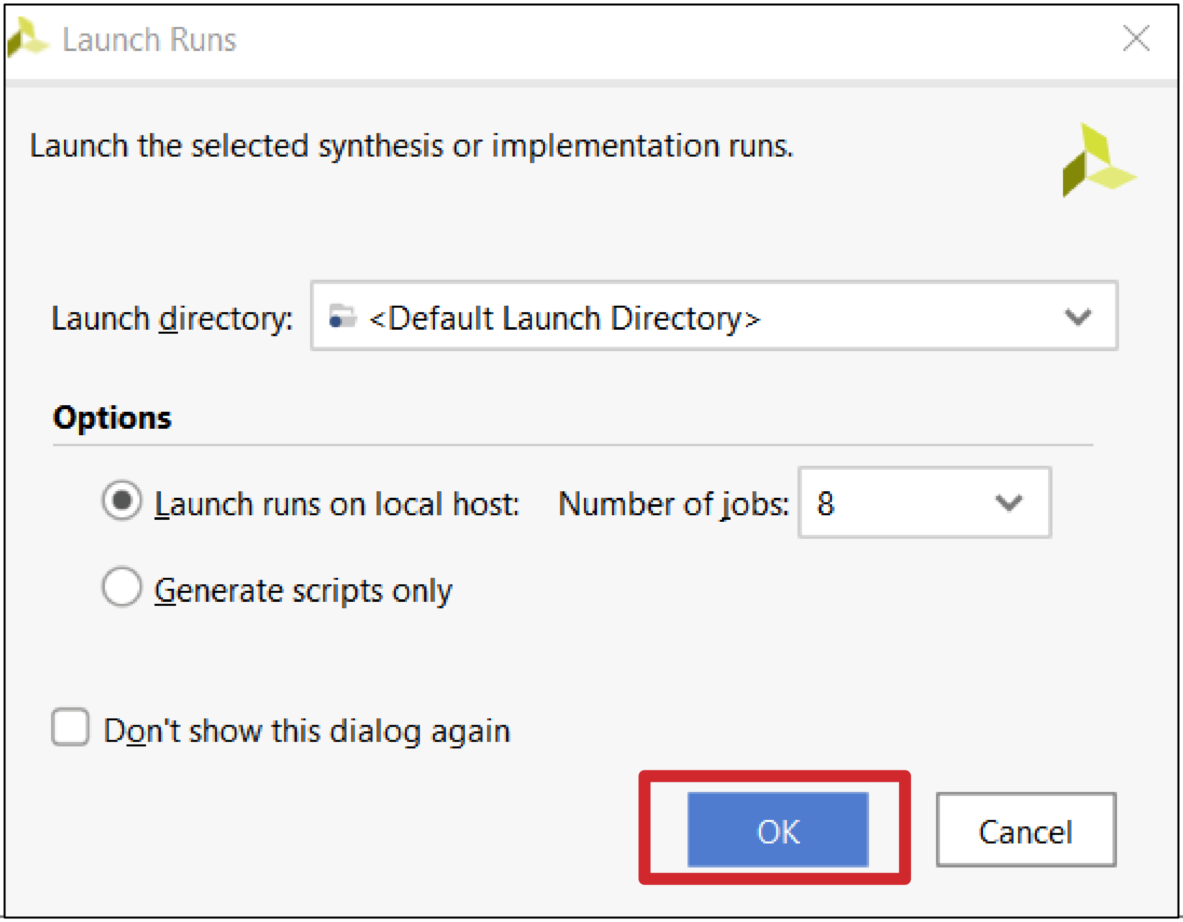
SYNTHESIS
Run Synthesis
Open Synthesized Design

IMPLEMENTATION
Run Implementation
Open Implemented Design

PROGRAM AND DEBUG

DESIGN EXAMPLE

- Vivado project
  - Logic synthesis



Launch Runs

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

Options

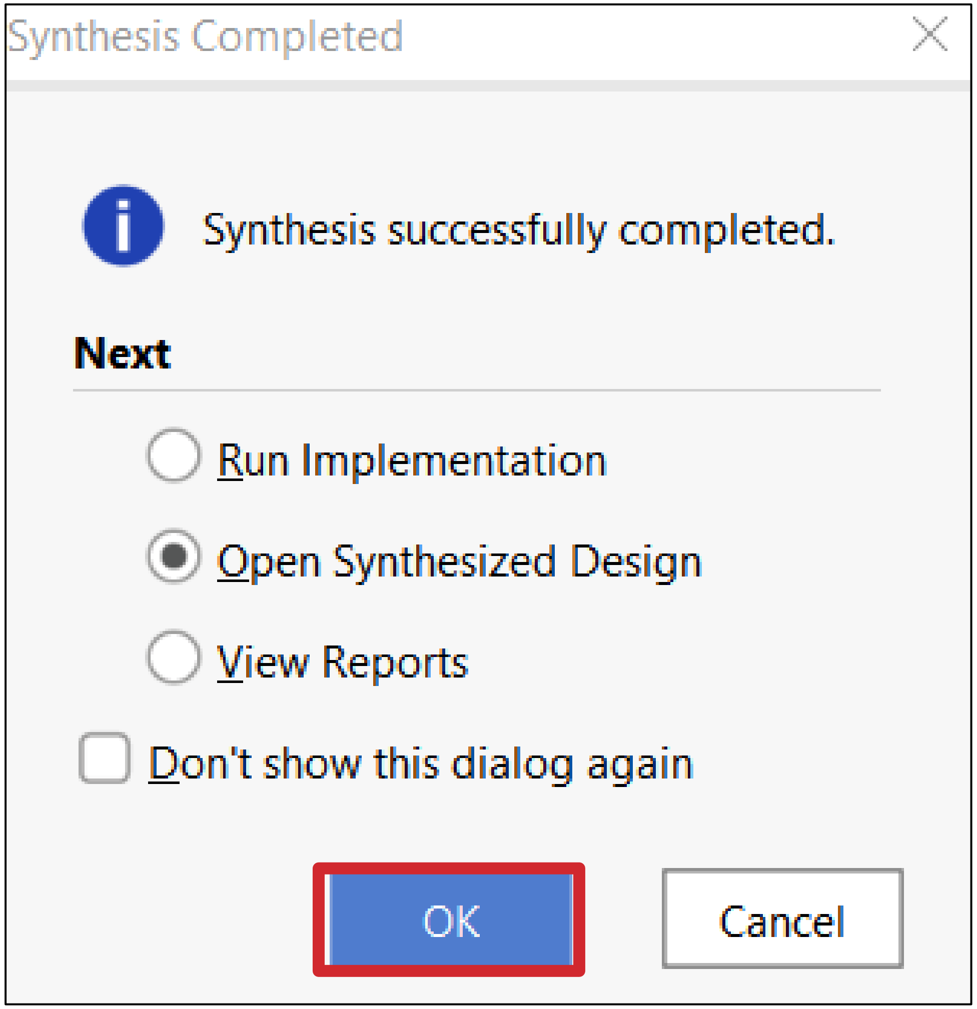
(●) Launch runs on local host: Number of jobs: 8

( ) Generate scripts only

☐ Don't show this dialog again

DESIGN EXAMPLE

- Vivado project
  - Logic synthesis


Synthesis Completed

Synthesis successfully completed.

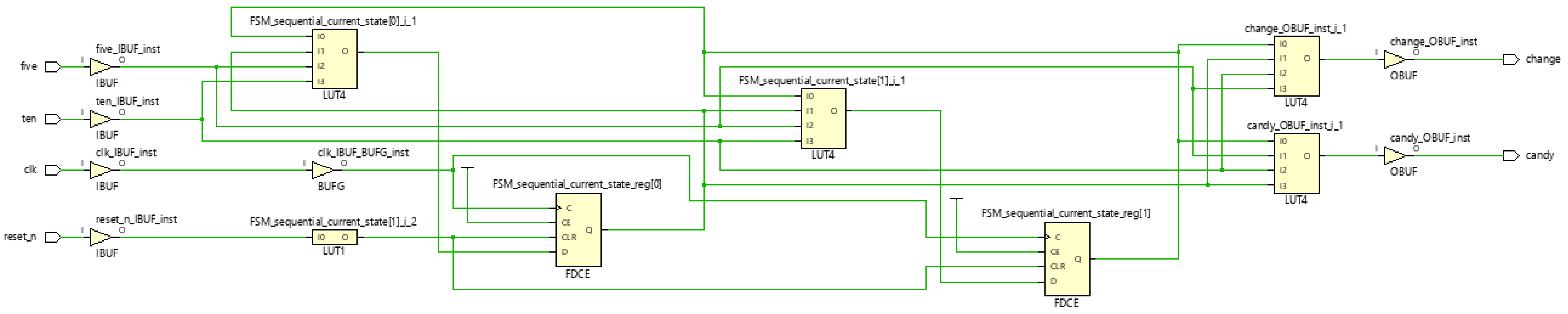
Next

- ( ) Run Implementation
- (●) Open Synthesized Design
- ( ) View Reports

☐ Don't show this dialog again

DESIGN EXAMPLE

- Vivado project
  - Logic synthesis (Schematic)



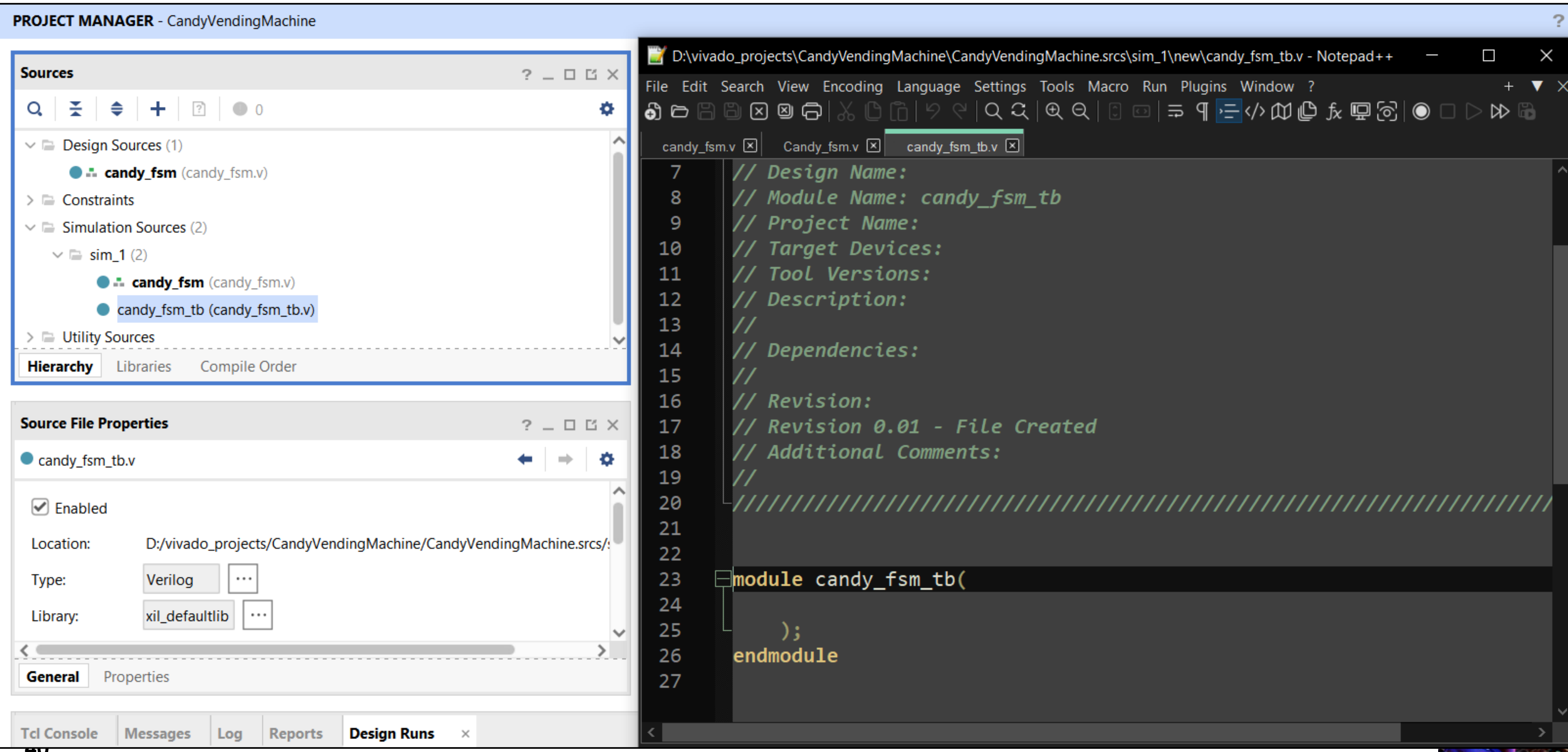
DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy FSM Testbench (Test Vectors)
    - Coin input sequence: five(5) -> ten(10) -> ten(10)

current_state	five	ten	next_state	candy	change
zero	0	0	zero	0	0
zero	1	0	five	0	0
five	0	1	fifteen	0	0
fifteen	0	1	zero	1	1

DESIGN EXAMPLE

- Vivado project
  - Creating a Testbench



PROJECT MANAGER - CandyVendingMachine

Sources

- Design Sources (1)
  - candy_fsm (candy_fsm.v)
- Constraints
- Simulation Sources (2)
  - sim_1 (2)
    - candy_fsm (candy_fsm.v)
    - candy_fsm_tb (candy_fsm_tb.v)
- Utility Sources

Source File Properties

candy_fsm_tb.v

Enabled

Location: D:/vivado_projects/CandyVendingMachine/CandyVendingMachine.srcs/

Type: Verilog

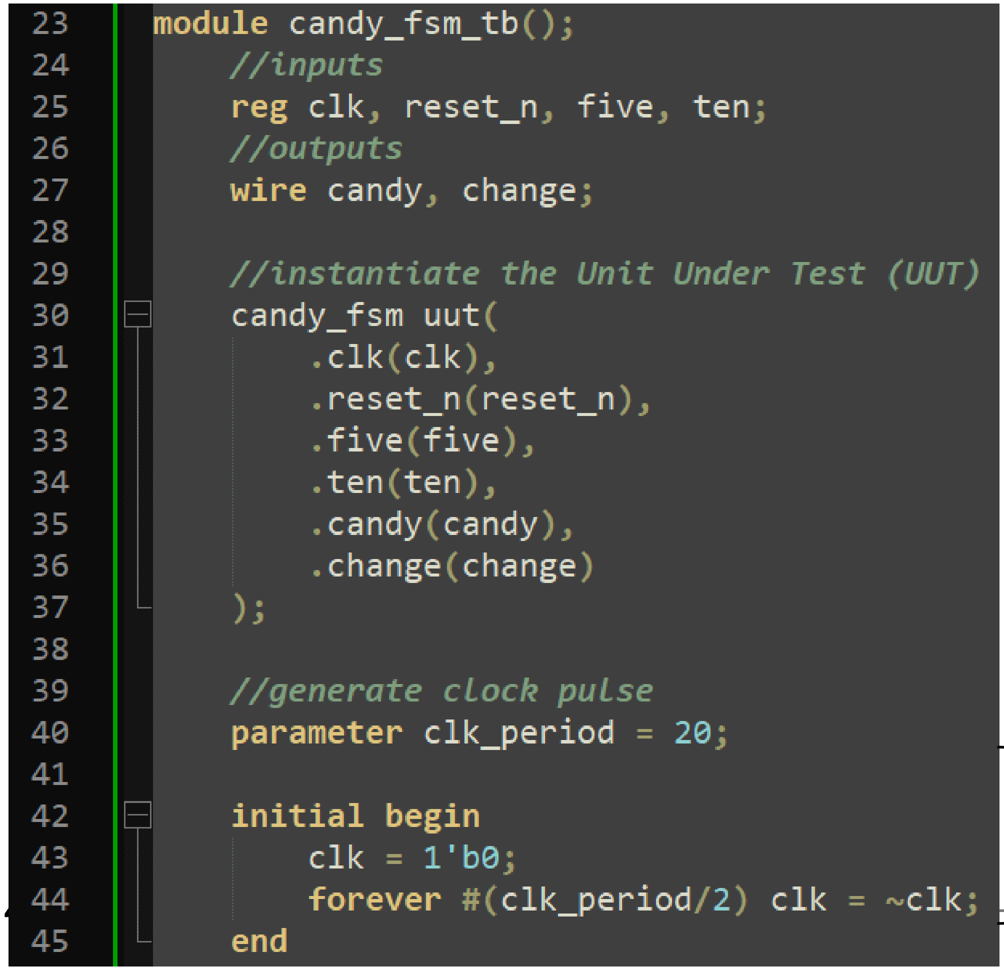
Library: xil_defaultlib

D:\vivado_projects\CandyVendingMachine\CandyVendingMachine.srcs\sim_1\new\candy_fsm_tb.v - Notepad++

```
// Design Name:
// Module Name: candy_fsm_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////


module candy_fsm_tb(

    );
endmodule
```

DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy FSM Testbench



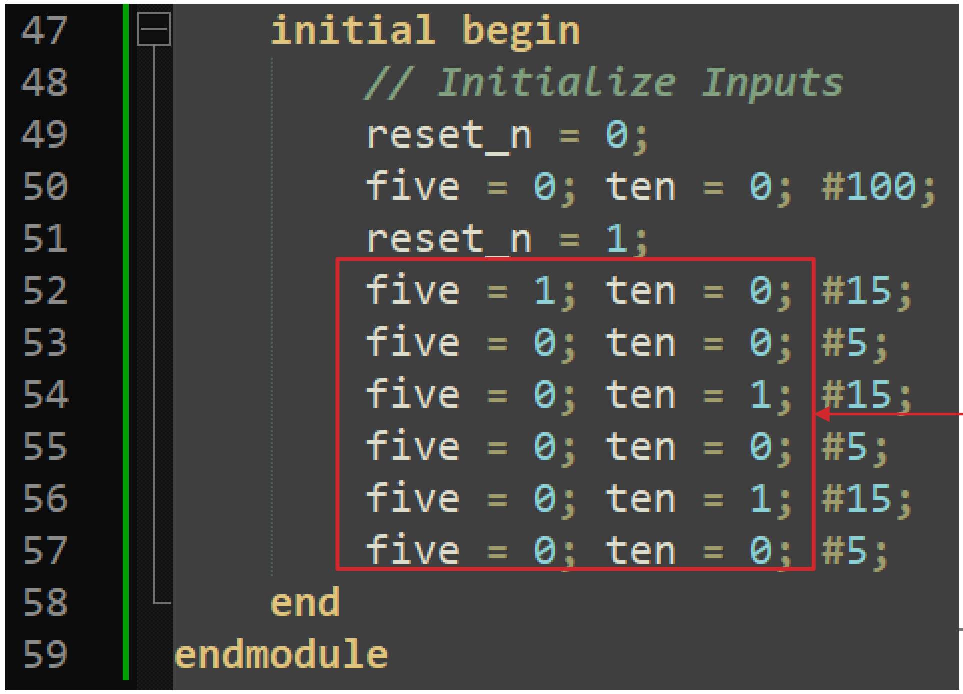
```
module candy_fsm_tb();
    //inputs
    reg clk, reset_n, five, ten;
    //outputs
    wire candy, change;

    //instantiate the Unit Under Test (UUT)
    candy_fsm uut(
        .clk(clk),
        .reset_n(reset_n),
        .five(five),
        .ten(ten),
        .candy(candy),
        .change(change)
    );

    //generate clock pulse
    parameter clk_period = 20;

    initial begin
        clk = 1'b0;
        forever #(clk_period/2) clk = ~clk;
    end
```

Generate clock pulse using 50Mhz frequency

DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy FSM Testbench

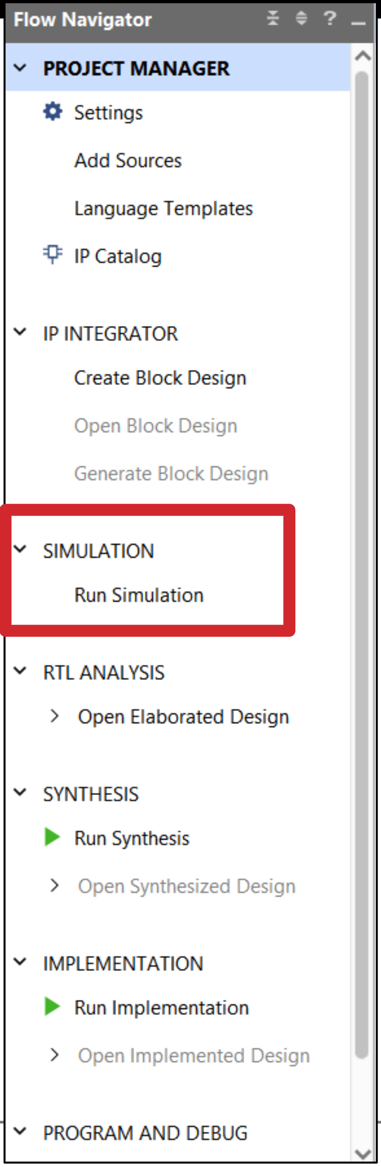
current_state	five	ten	next_state	candy	change
zero	0	0	zero	0	0
zero	1	0	five	0	0
five	0	1	fifteen	0	0
fifteen	0	1	zero	1	1



```
    initial begin
        // Initialize Inputs
        reset_n = 0;
        five = 0; ten = 0; #100;
        reset_n = 1;
        five = 1; ten = 0; #15;
        five = 0; ten = 0; #5;
        five = 0; ten = 1; #15;
        five = 0; ten = 0; #5;
        five = 0; ten = 1; #15;
        five = 0; ten = 0; #5;
    end
endmodule
```

DESIGN EXAMPLE

- Vivado project
  - Running Simulation


Flow Navigator

PROJECT MANAGER
- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR
- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION
- Run Simulation

RTL ANALYSIS
- Open Elaborated Design

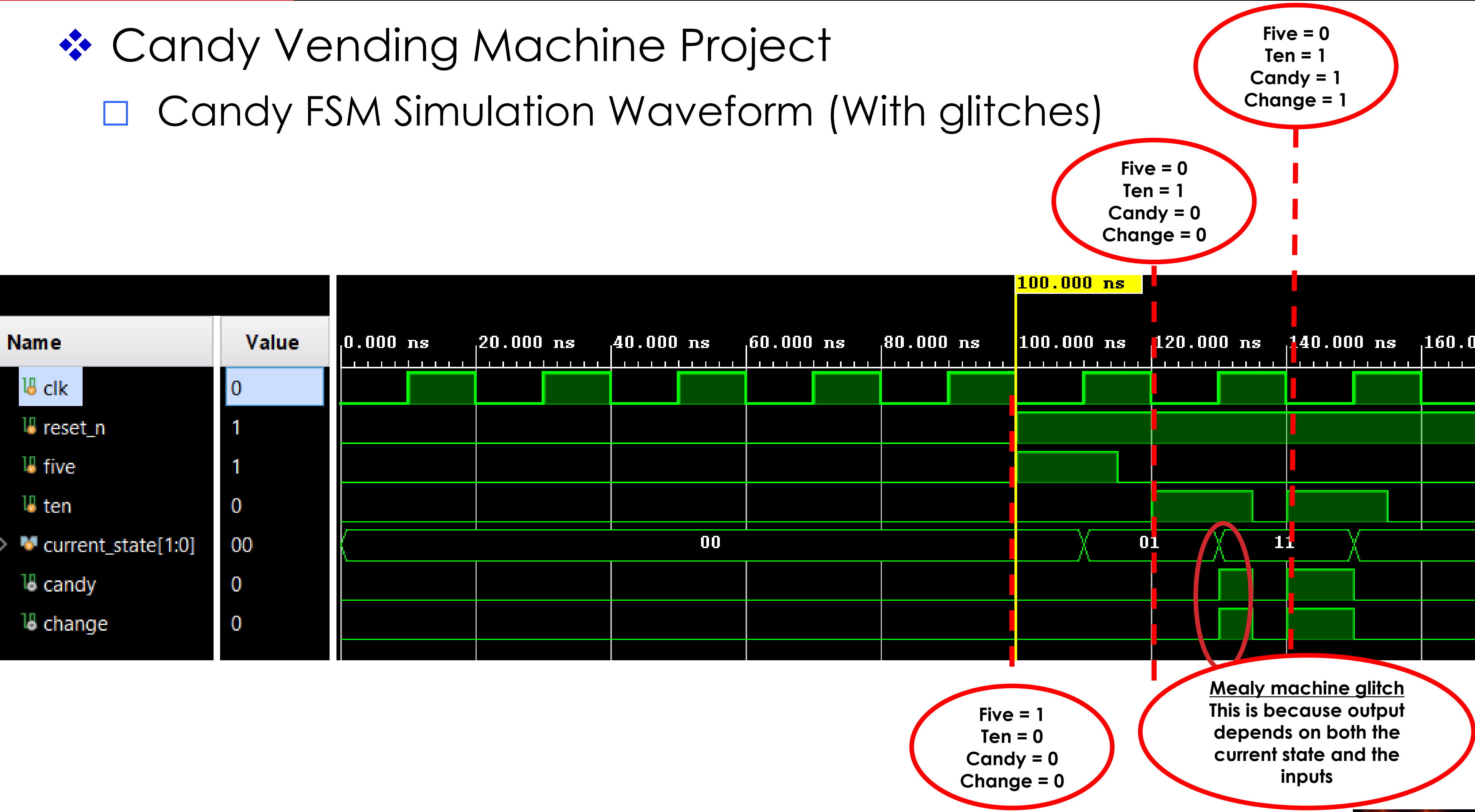
SYNTHESIS
- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION
- Run Implementation
- Open Implemented Design

PROGRAM AND DEBUG

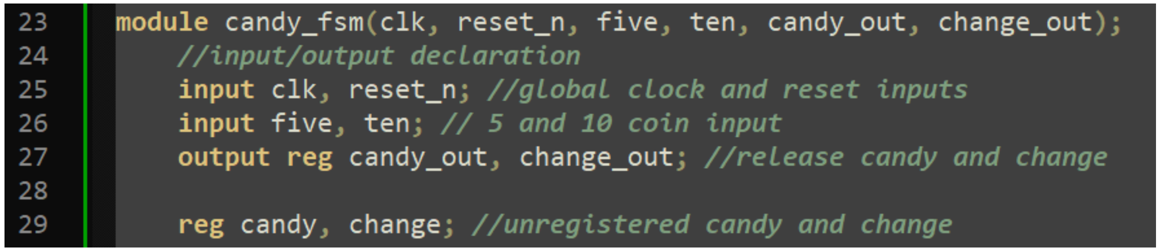
DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy FSM Simulation Waveform (With glitches)

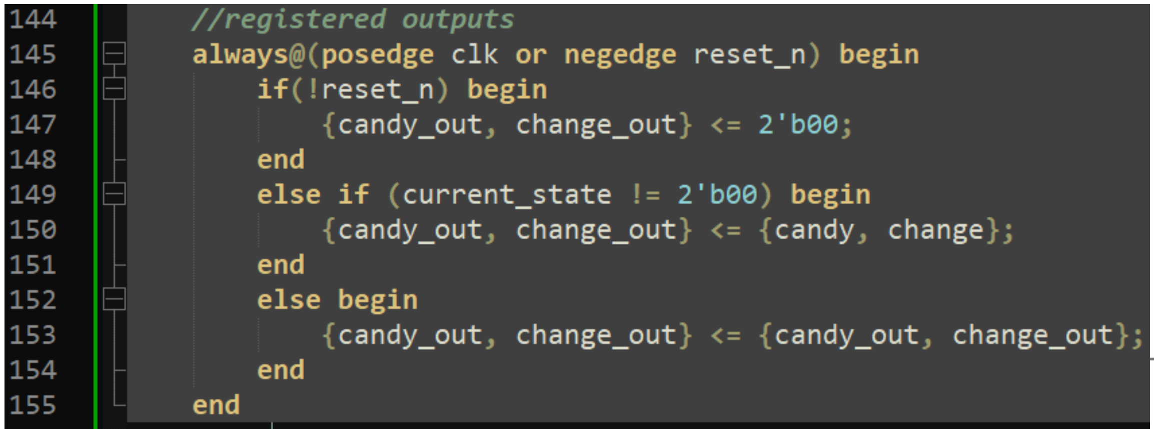


DESIGN EXAMPLE

- Candy Vending Machine Project
  - Solving Mealy Glitch
    - One solution is by using registered outputs as shown in the code



```
module candy_fsm(clk, reset_n, five, ten, candy_out, change_out);
    //input/output declaration
    input clk, reset_n; //global clock and reset inputs
    input five, ten; // 5 and 10 coin input
    output reg candy_out, change_out; //release candy and change

    reg candy, change; //unregistered candy and change
```

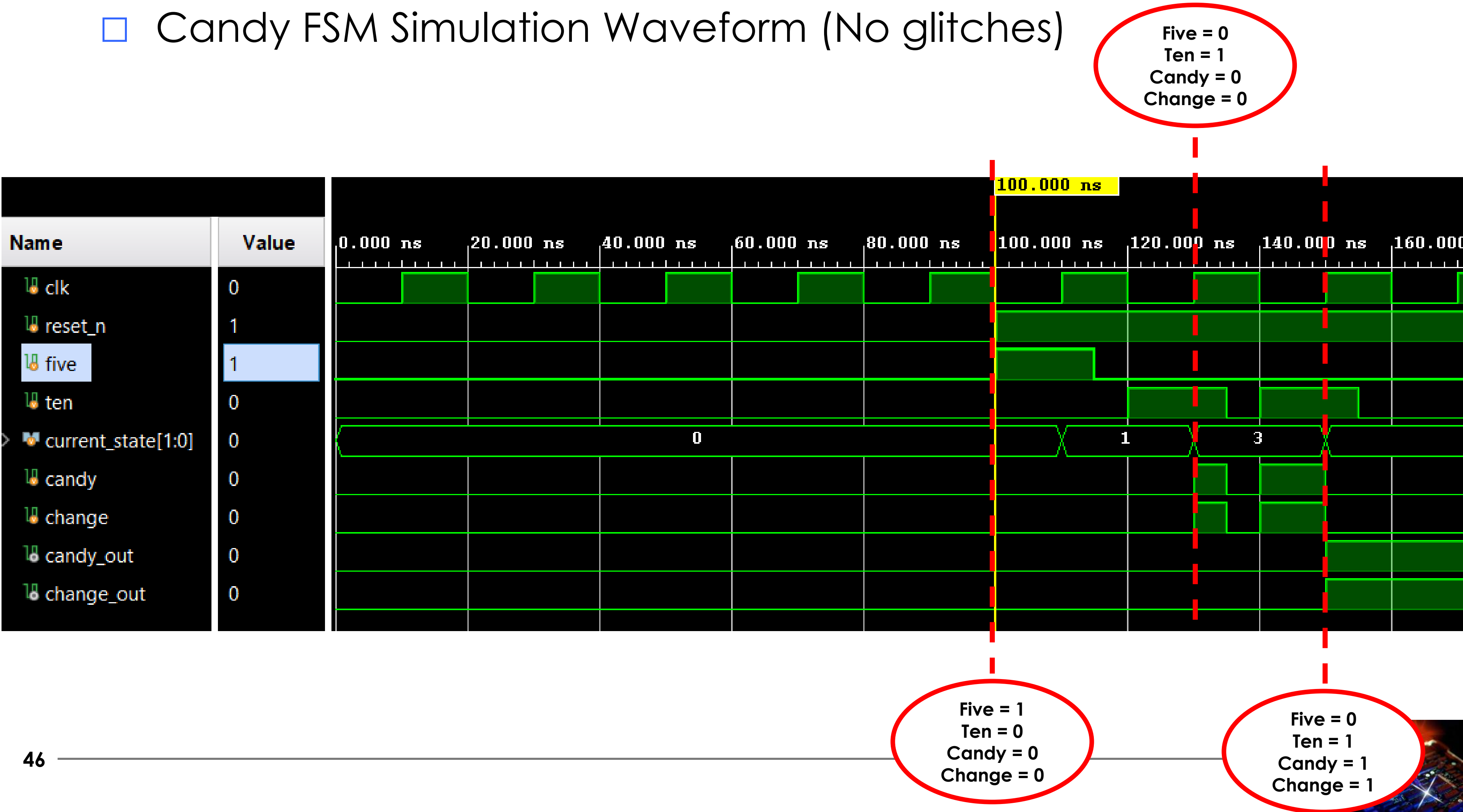


```
    //registered outputs
    always@(posedge clk or negedge reset_n) begin
        if(!reset_n) begin
            {candy_out, change_out} <= 2'b00;
        end
        else if (current_state != 2'b00) begin
            {candy_out, change_out} <= {candy, change};
        end
        else begin
            {candy_out, change_out} <= {candy_out, change_out};
        end
    end
```

The outputs (candy and change) are registered at the rising edge of the clock to prevent glitches

DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy FSM Simulation Waveform (No glitches)

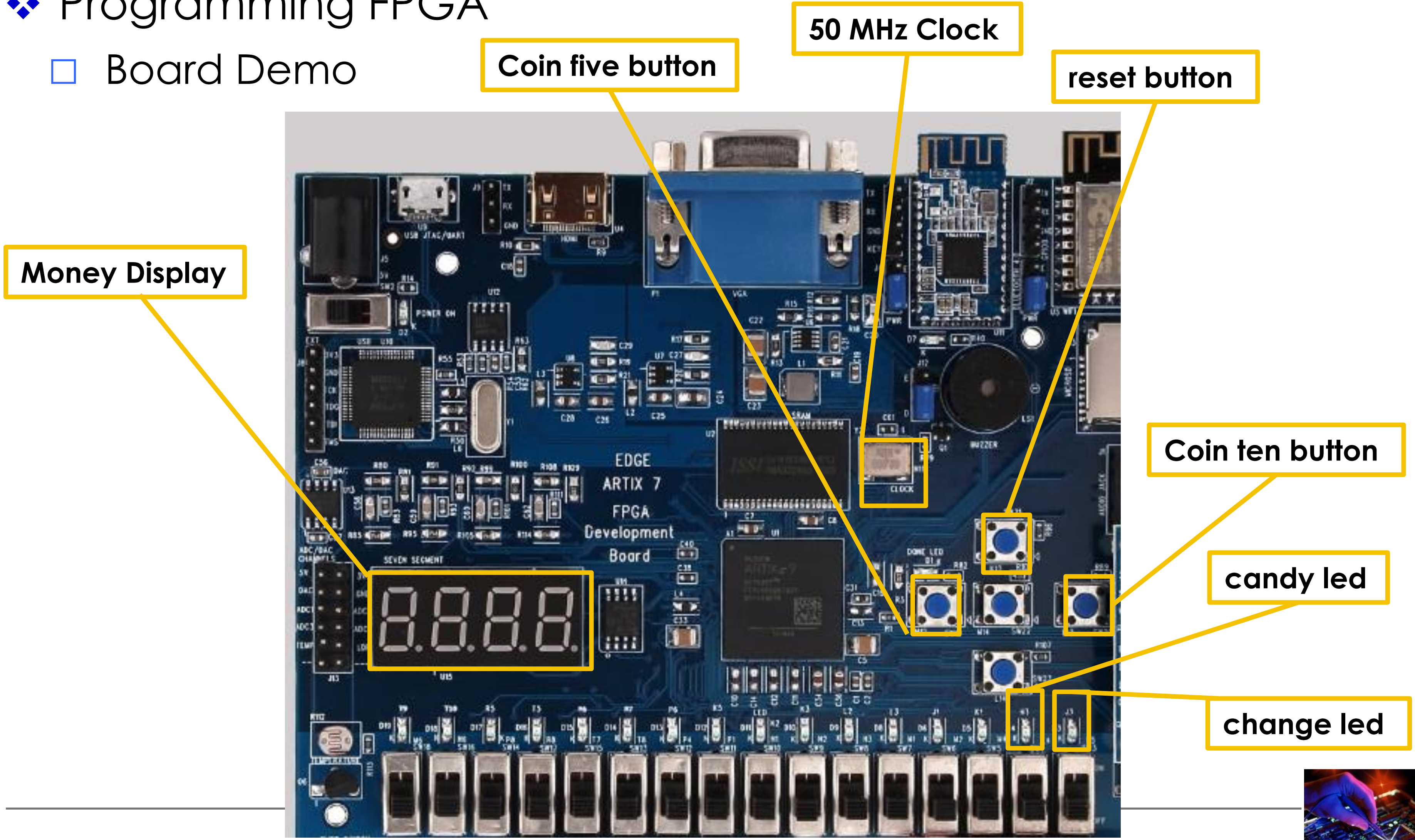


DESIGN EXAMPLE

- Candy Vending Machine Project: Exercise
  - Design and Simulate Testbenches for the input sequences below:
    - Coin input sequence: five(5) -> five(5) -> five(5) -> five(5)
    - Coin input sequence: five(5) -> ten(10) -> five(5)
    - Coin input sequence: ten(10) -> five(5) -> five(5)
    - Coin input sequence: ten(10) -> ten(10)
    - Coin input sequence: ten(10) -> five(5) -> ten(10)

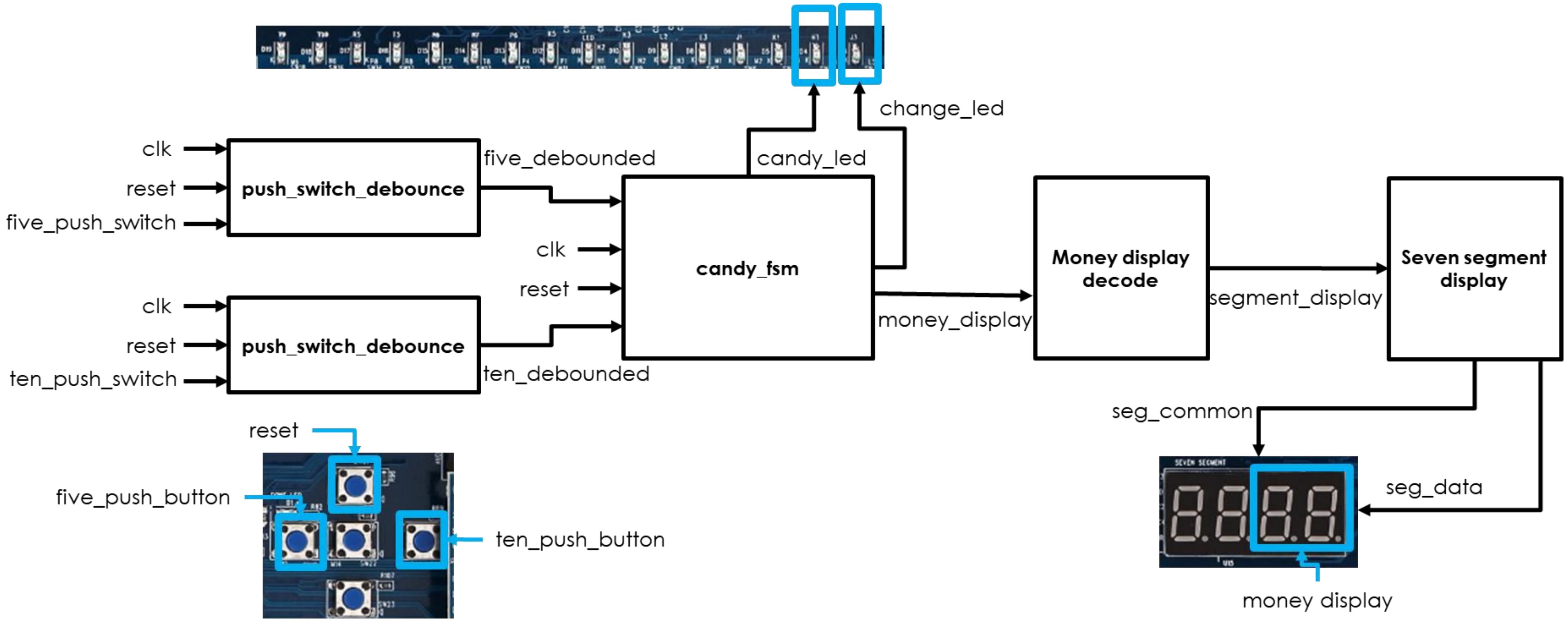
DESIGN EXAMPLE

- Programming FPGA
  - Board Demo



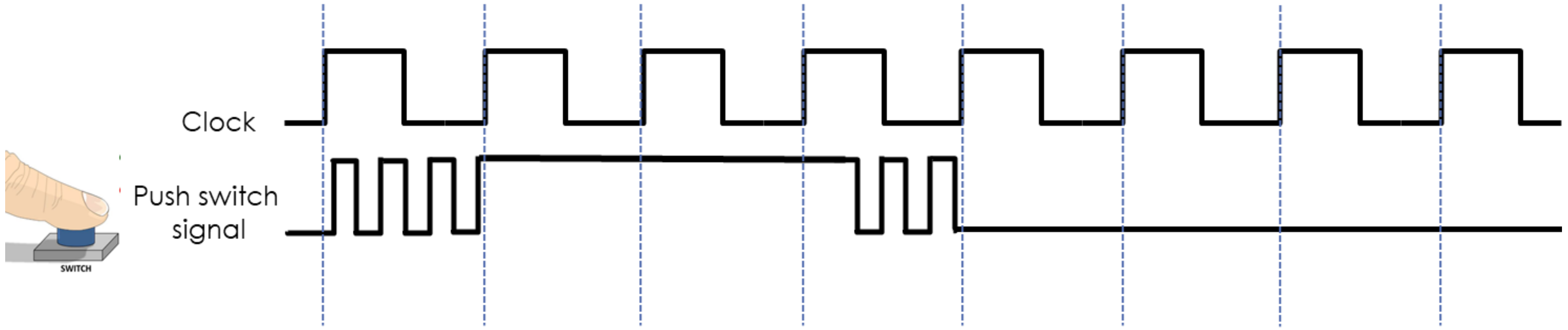
DESIGN EXAMPLE

- Candy Vending Machine Project
  - Modules of Candy Vending Machine



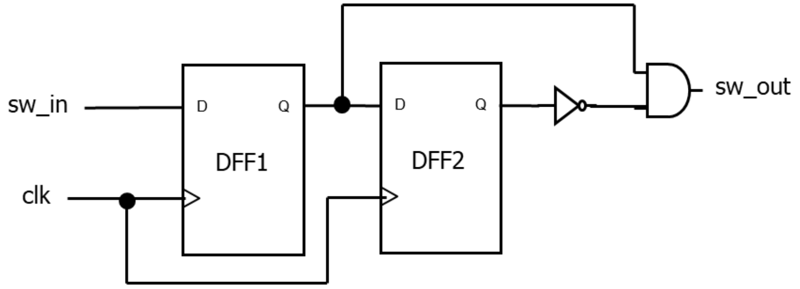
DESIGN EXAMPLE

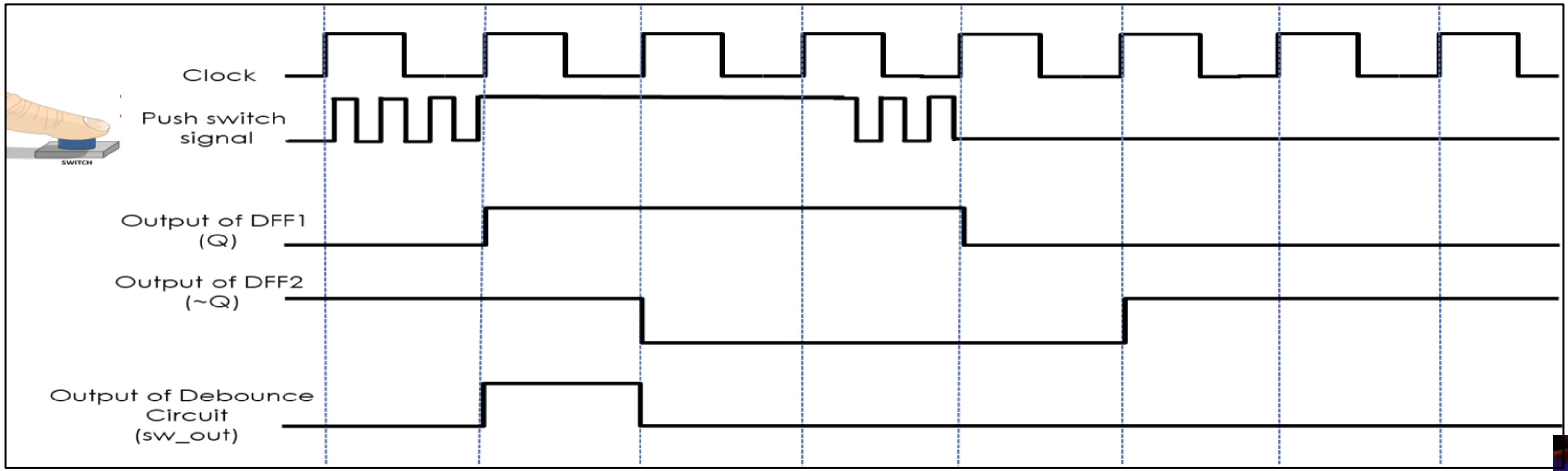
- Candy Vending Machine Project
  - push_switch_debounce Module
    - As shown in the figure below, when a button on FPGA is pressed and released, there are many unexpected up-and-down bounces in the push button signal



DESIGN EXAMPLE

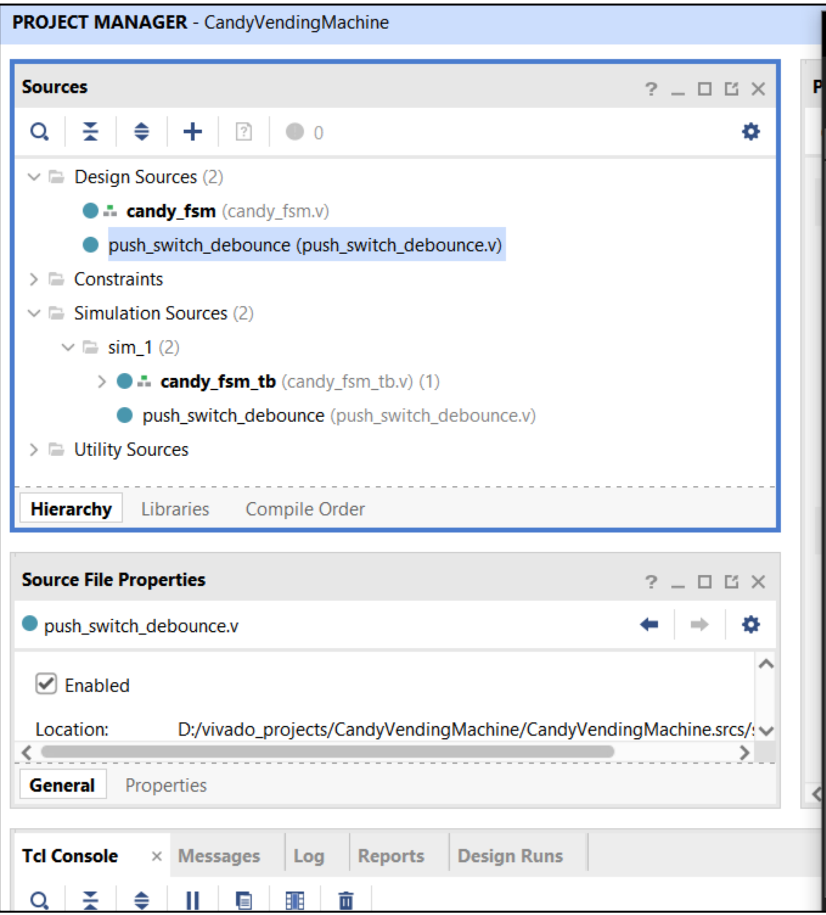
- Candy Vending Machine Project
  - push_switch_debounce Module
    - The debouncing circuit (using two D flip-flops) only generates a single pulse with a period of the slow clock without bouncing as we expected





DESIGN EXAMPLE

- Candy Vending Machine Project
  - push_switch_debounce Verilog code
    - Module name: push_switch_debounce.v


PROJECT MANAGER - CandyVendingMachine

Sources

- Design Sources (2)
  - candy_fsm (candy_fsm.v)
  - push_switch_debounce (push_switch_debounce.v)
- Constraints
- Simulation Sources (2)
  - sim_1 (2)
    - candy_fsm_tb (candy_fsm_tb.v) (1)
    - push_switch_debounce (push_switch_debounce.v)
- Utility Sources

Hierarchy | Libraries | Compile Order

Source File Properties

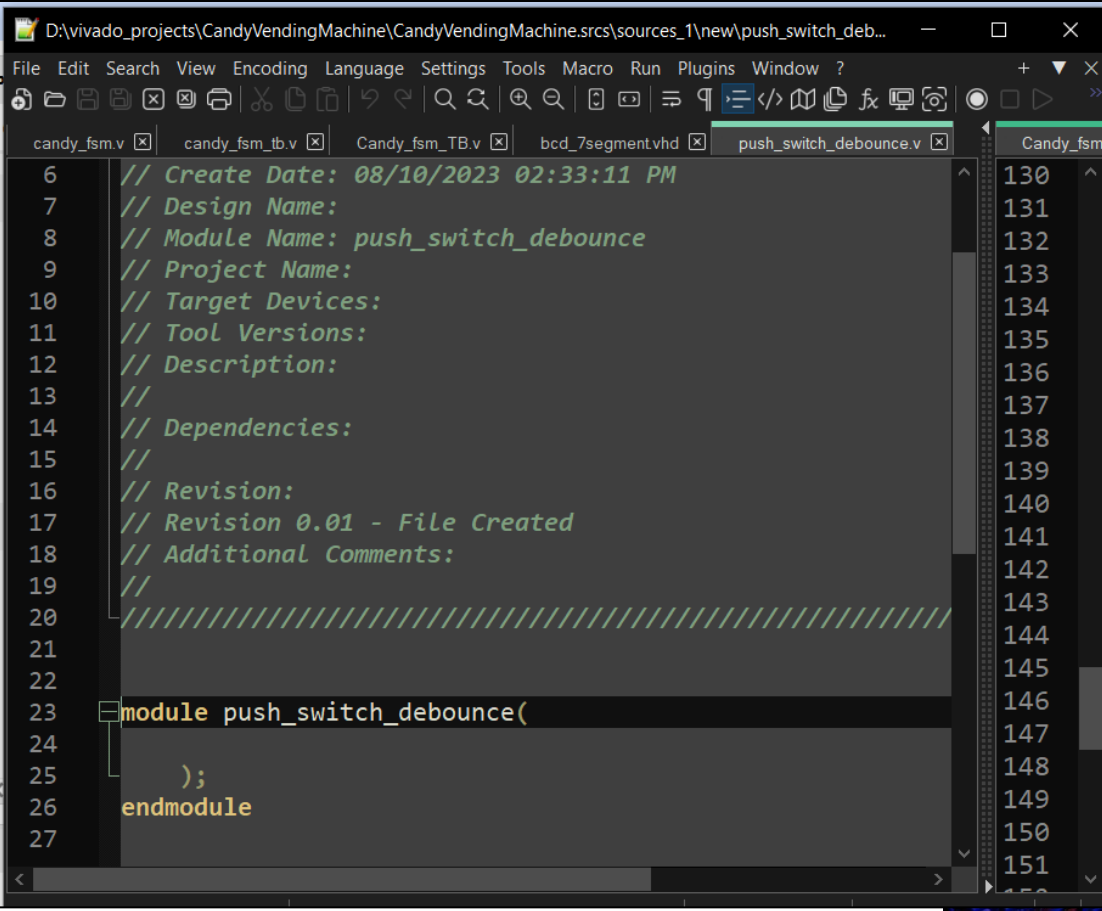
push_switch_debounce.v

Enabled ✓

Location: D:/vivado_projects/CandyVendingMachine/CandyVendingMachine.srcs/:

General | Properties

Tcl Console | Messages | Log | Reports | Design Runs


D:\vivado_projects\CandyVendingMachine\CandyVendingMachine.srcs\sources_1\new\push_switch_deb...

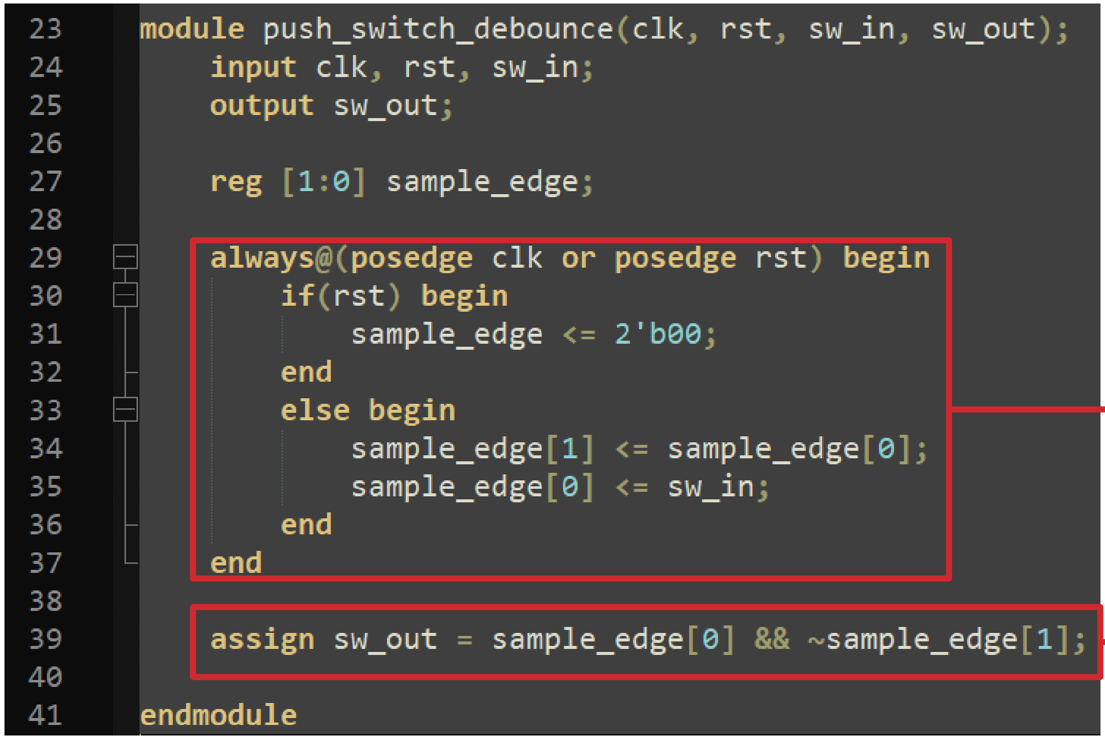
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

candy_fsm.v | candy_fsm_tb.v | Candy_fsm_TB.v | bcd_7segment.vhd | push_switch_debounce.v | Candy_fsm

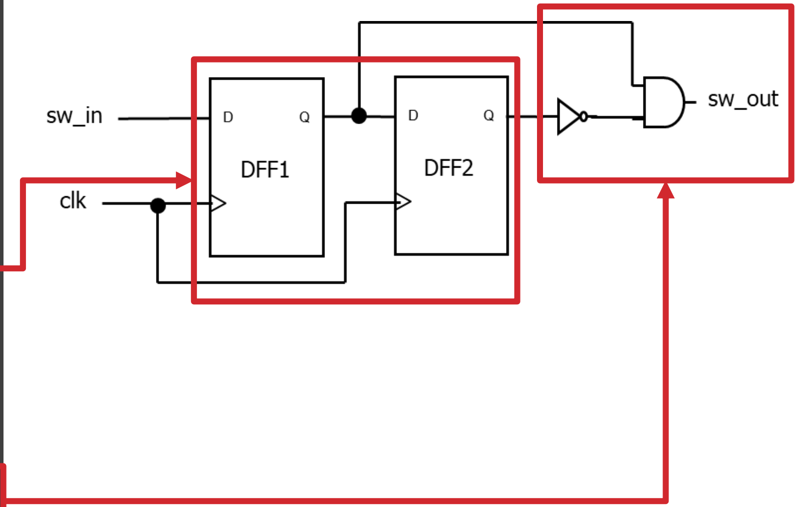
```
// Create Date: 08/10/2023 02:33:11 PM
// Design Name:
// Module Name: push_switch_debounce
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
///////////////////////////////////////////////////////////


module push_switch_debounce(

    );
endmodule
```

DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Verilog Code for Push Switch Debounce circuit

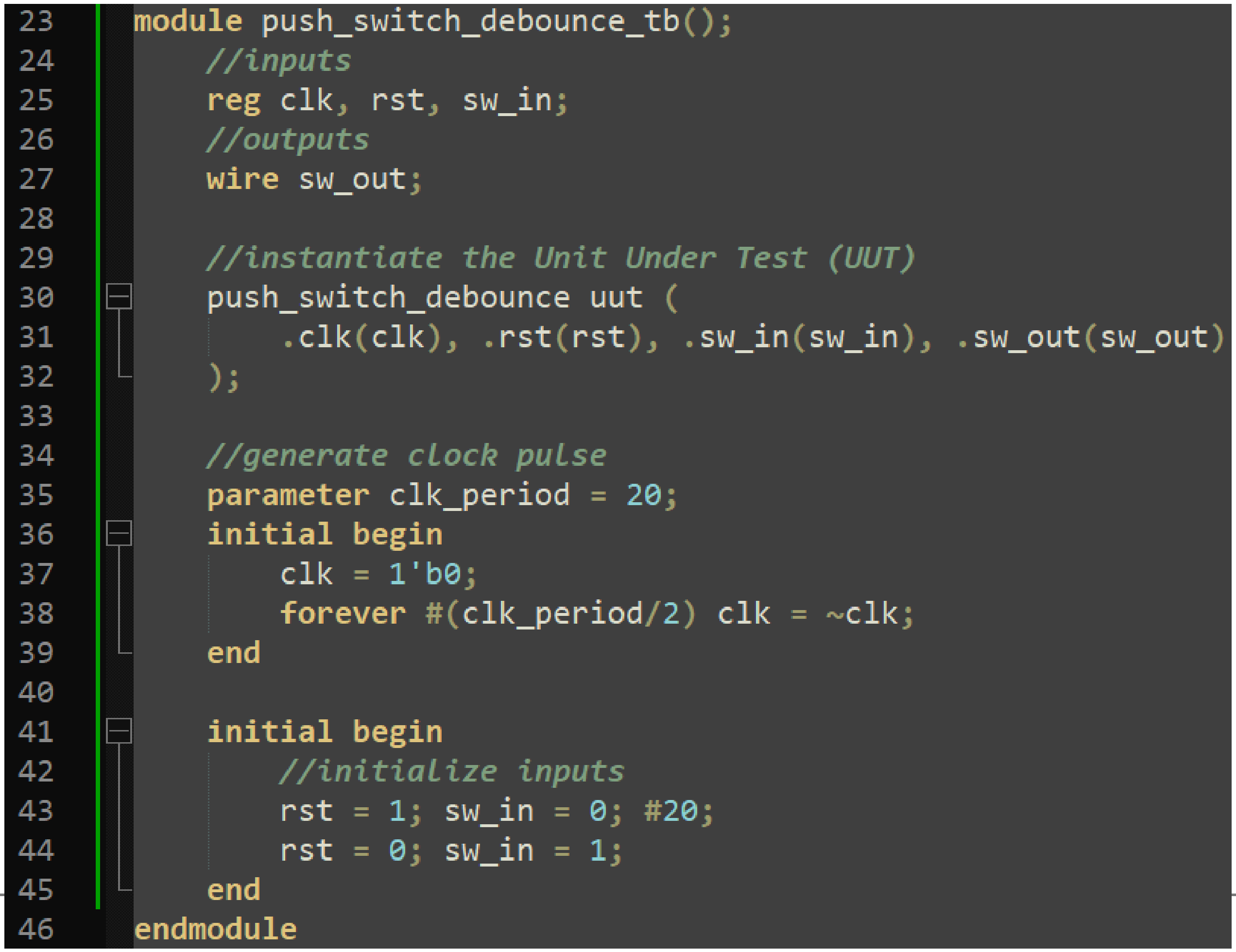


```verilog
module push_switch_debounce(clk, rst, sw_in, sw_out);
    input clk, rst, sw_in;
    output sw_out;

    reg [1:0] sample_edge;

    always@(posedge clk or posedge rst) begin
        if(rst) begin
            sample_edge <= 2'b00;
        end
        else begin
            sample_edge[1] <= sample_edge[0];
            sample_edge[0] <= sw_in;
        end
    end

    assign sw_out = sample_edge[0] && ~sample_edge[1];

endmodule
```



DESIGN EXAMPLE

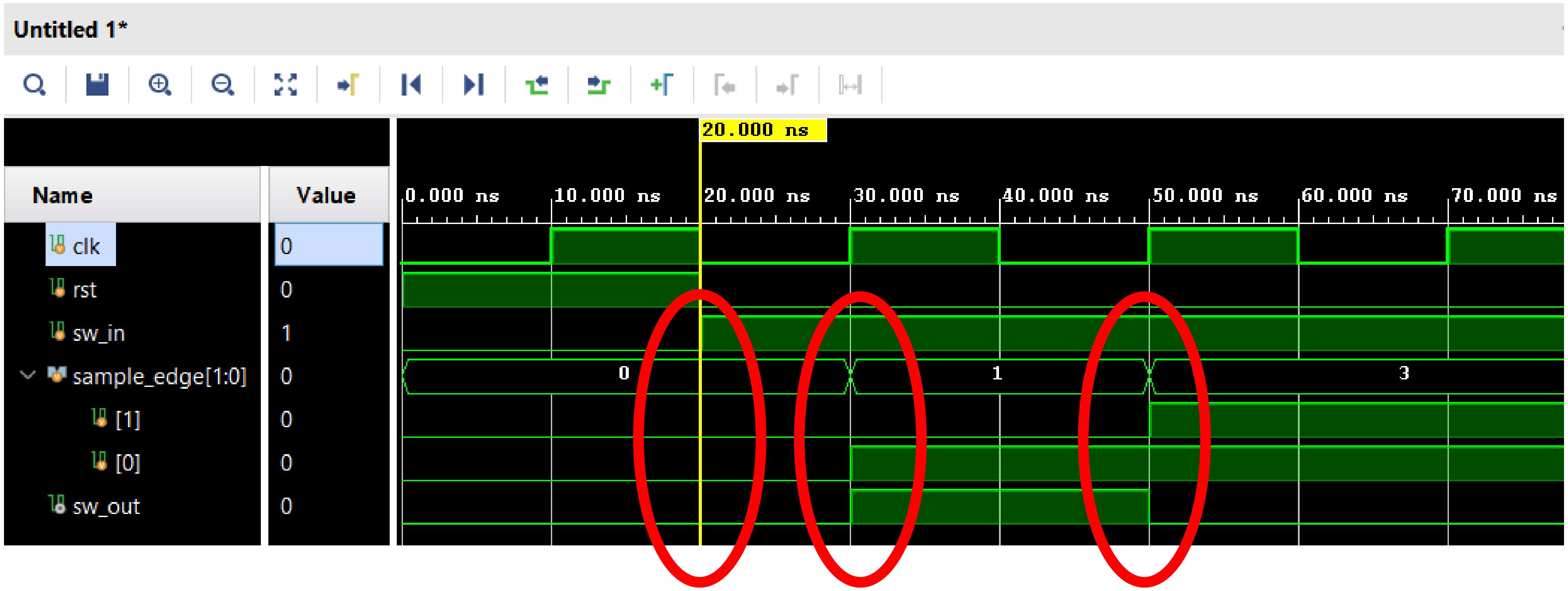
- Candy Vending Machine Project Project
  - Push Switch Debounce circuit Verilog Testbench Code



```
module push_switch_debounce_tb();
    //inputs
    reg clk, rst, sw_in;
    //outputs
    wire sw_out;

    //instantiate the Unit Under Test (UUT)
    push_switch_debounce uut (
        .clk(clk), .rst(rst), .sw_in(sw_in), .sw_out(sw_out)
    );

    //generate clock pulse
    parameter clk_period = 20;
    initial begin
        clk = 1'b0;
        forever #(clk_period/2) clk = ~clk;
    end

    initial begin
        //initialize inputs
        rst = 1; sw_in = 0; #20;
        rst = 0; sw_in = 1;
    end
endmodule
```

DESIGN EXAMPLE

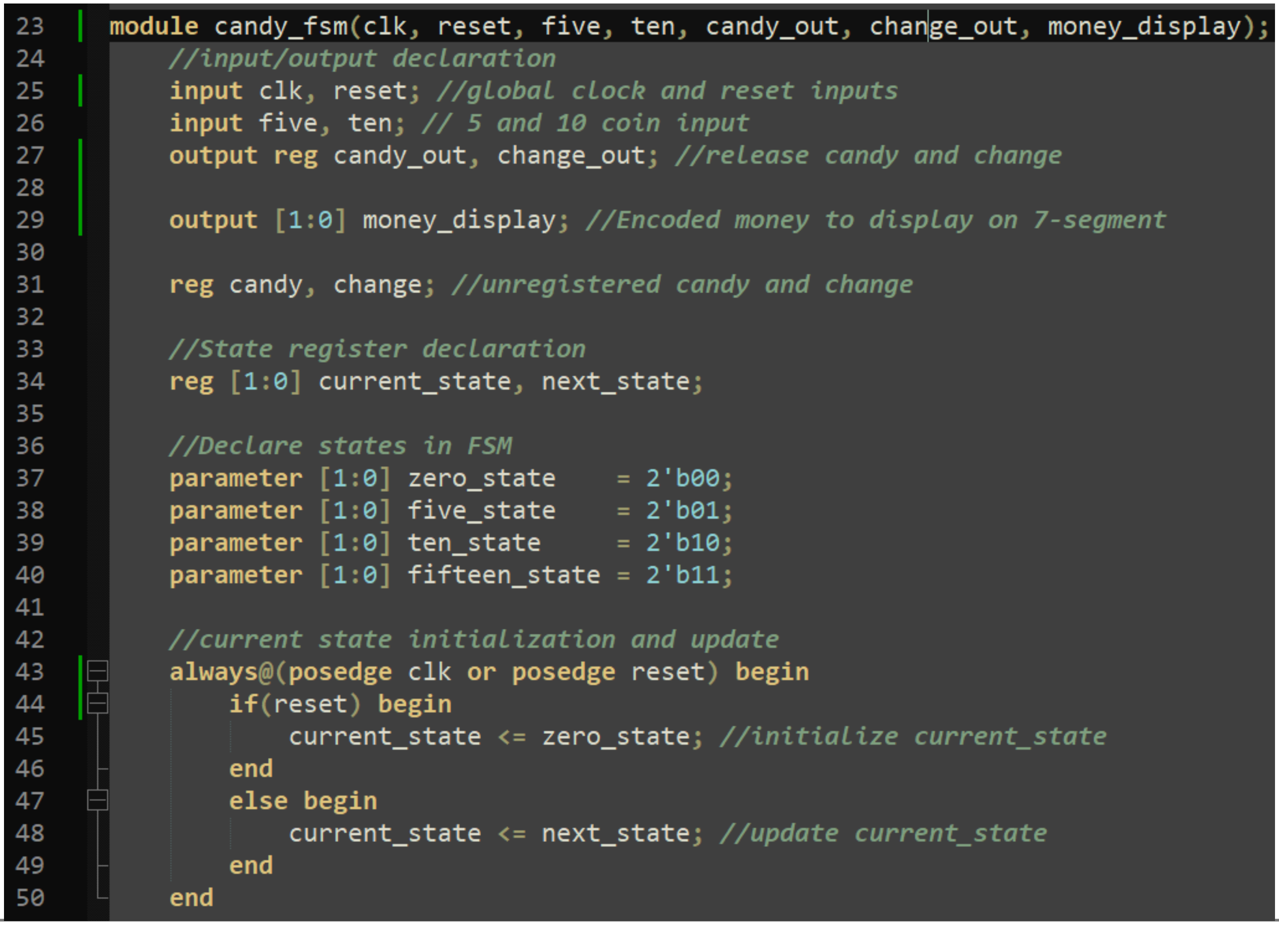
- Candy Vending Machine Project
  - Push Switch Debounce circuit Verilog Simulation Waveform



DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Candy FSM Verilog Code (Modified)

1

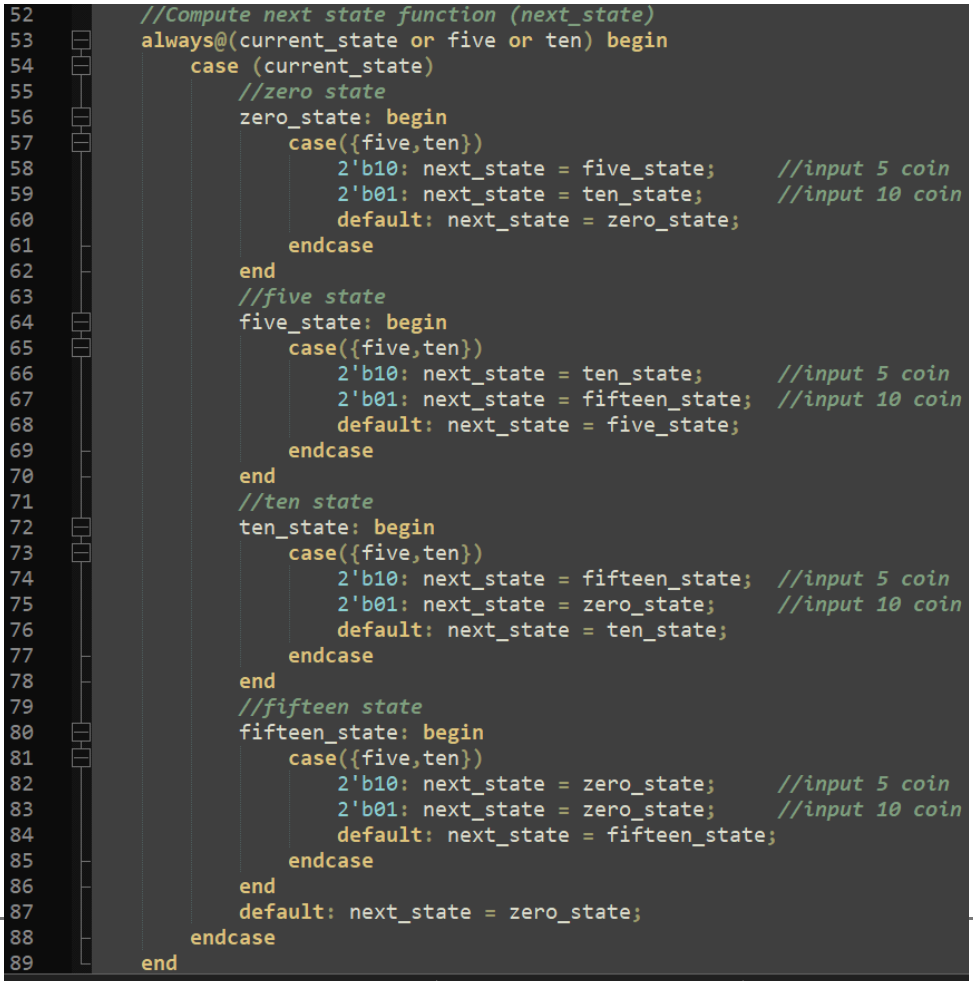


```
module candy_fsm(clk, reset, five, ten, candy_out, change_out, money_display);
    //input/output declaration
    input clk, reset; //global clock and reset inputs
    input five, ten; // 5 and 10 coin input
    output reg candy_out, change_out; //release candy and change

    output [1:0] money_display; //Encoded money to display on 7-segment

    reg candy, change; //unregistered candy and change

    //State register declaration
    reg [1:0] current_state, next_state;

    //Declare states in FSM
    parameter [1:0] zero_state    = 2'b00;
    parameter [1:0] five_state    = 2'b01;
    parameter [1:0] ten_state     = 2'b10;
    parameter [1:0] fifteen_state = 2'b11;

    //current state initialization and update
    always@(posedge clk or posedge reset) begin
        if(reset) begin
            current_state <= zero_state; //initialize current_state
        end
        else begin
            current_state <= next_state; //update current_state
        end
    end
```

DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Candy FSM Verilog Code (Modified)

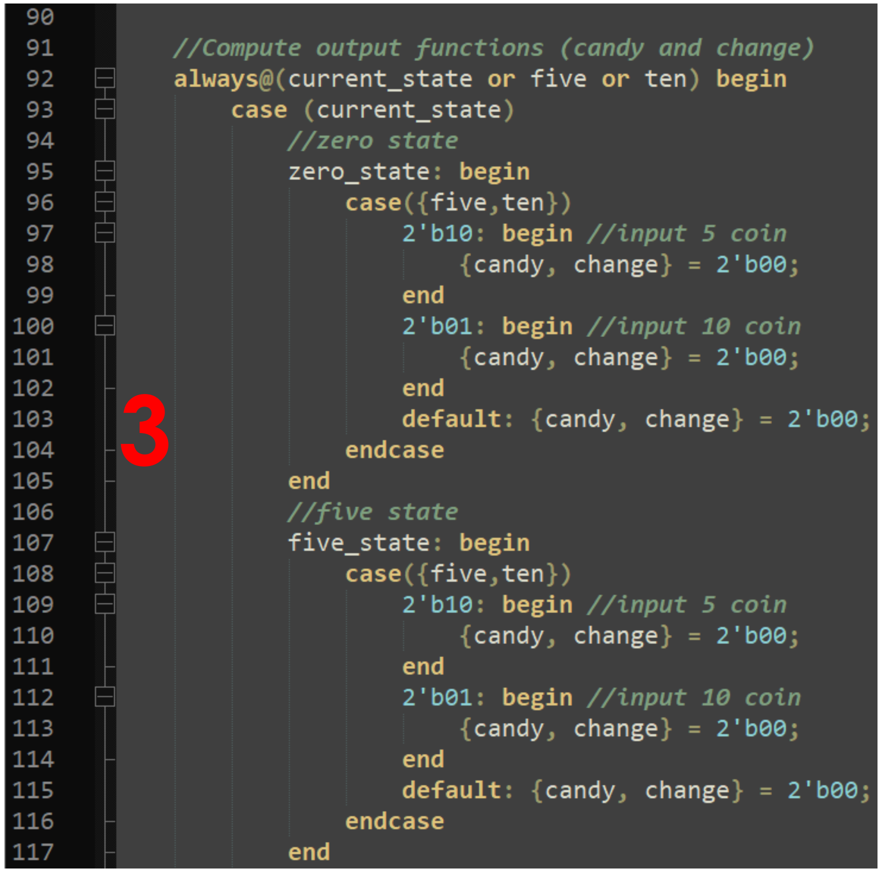
2



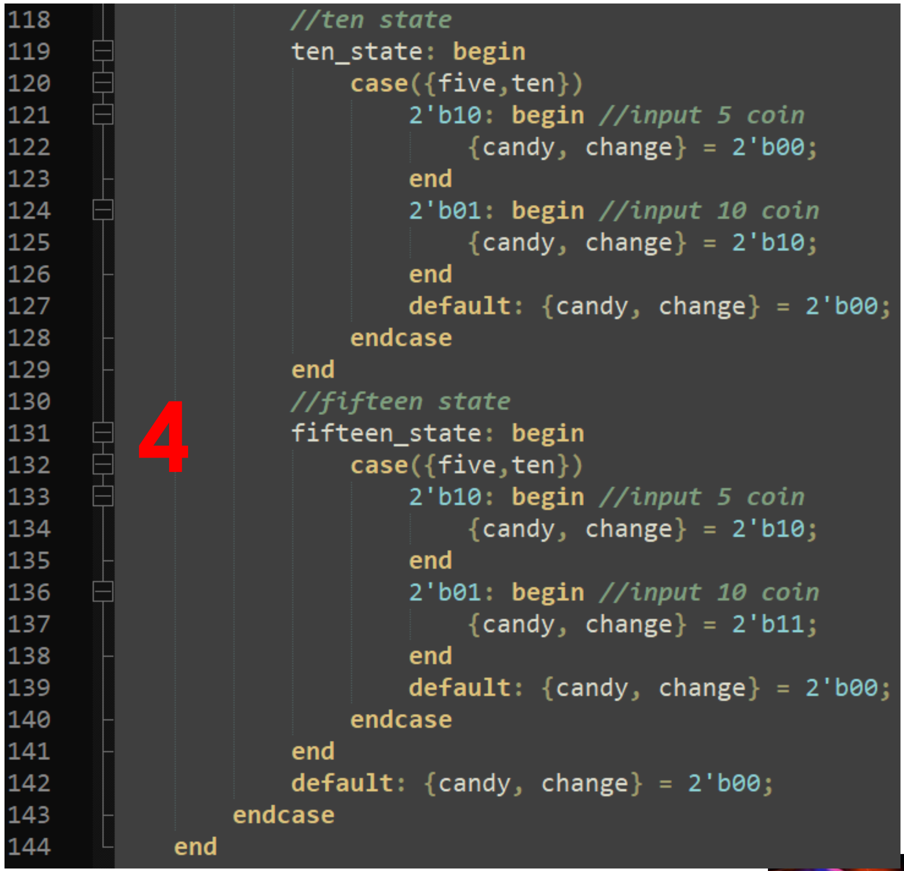
```verilog
//Compute next state function (next_state)
always@(current_state or five or ten) begin
    case (current_state)
        //zero state
        zero_state: begin
            case({five,ten})
                2'b10: next_state = five_state;     //input 5 coin
                2'b01: next_state = ten_state;      //input 10 coin
                default: next_state = zero_state;
            endcase
        end
        //five state
        five_state: begin
            case({five,ten})
                2'b10: next_state = ten_state;      //input 5 coin
                2'b01: next_state = fifteen_state;  //input 10 coin
                default: next_state = five_state;
            endcase
        end
        //ten state
        ten_state: begin
            case({five,ten})
                2'b10: next_state = fifteen_state;  //input 5 coin
                2'b01: next_state = zero_state;     //input 10 coin
                default: next_state = ten_state;
            endcase
        end
        //fifteen state
        fifteen_state: begin
            case({five,ten})
                2'b10: next_state = zero_state;     //input 5 coin
                2'b01: next_state = zero_state;     //input 10 coin
                default: next_state = fifteen_state;
            endcase
        end
        default: next_state = zero_state;
    endcase
end
```

DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Candy FSM Verilog Code (Modified)



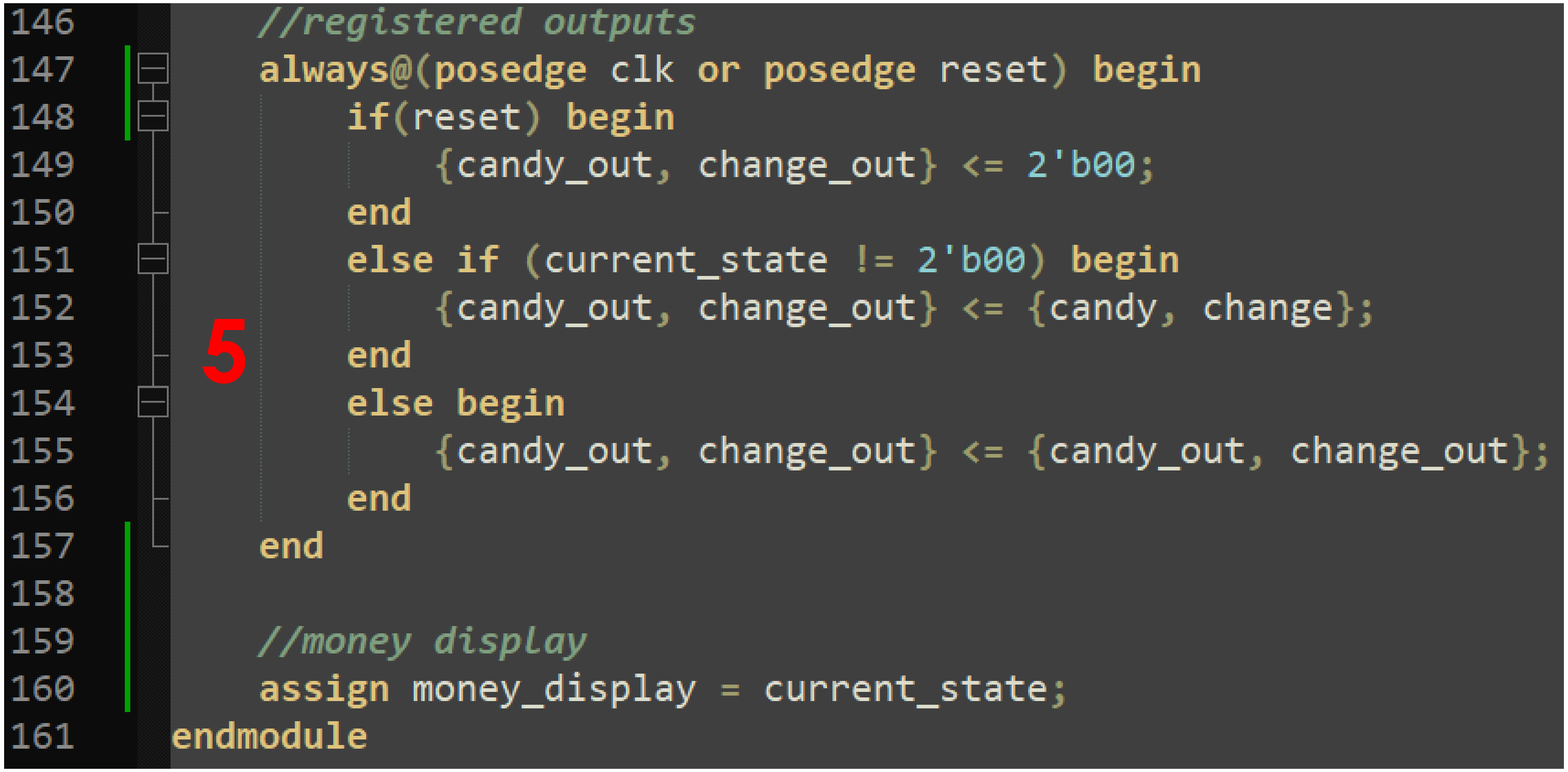
```verilog

    //Compute output functions (candy and change)
    always@(current_state or five or ten) begin
        case (current_state)
            //zero state
            zero_state: begin
                case({five,ten})
                    2'b10: begin //input 5 coin
                        {candy, change} = 2'b00;
                    end
                    2'b01: begin //input 10 coin
                        {candy, change} = 2'b00;
                    end
                    default: {candy, change} = 2'b00;
                endcase
            end
            //five state
            five_state: begin
                case({five,ten})
                    2'b10: begin //input 5 coin
                        {candy, change} = 2'b00;
                    end
                    2'b01: begin //input 10 coin
                        {candy, change} = 2'b00;
                    end
                    default: {candy, change} = 2'b00;
                endcase
            end
```



```verilog
            //ten state
            ten_state: begin
                case({five,ten})
                    2'b10: begin //input 5 coin
                        {candy, change} = 2'b00;
                    end
                    2'b01: begin //input 10 coin
                        {candy, change} = 2'b10;
                    end
                    default: {candy, change} = 2'b00;
                endcase
            end
            //fifteen state
            fifteen_state: begin
                case({five,ten})
                    2'b10: begin //input 5 coin
                        {candy, change} = 2'b10;
                    end
                    2'b01: begin //input 10 coin
                        {candy, change} = 2'b11;
                    end
                    default: {candy, change} = 2'b00;
                endcase
            end
            default: {candy, change} = 2'b00;
        endcase
    end
```

DESIGN EXAMPLE

- Candy Vending Machine Project Project
  - Candy FSM Verilog Code (Modified)

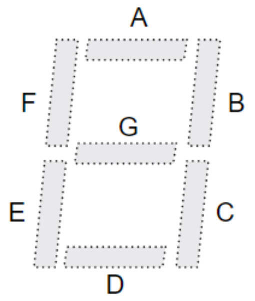


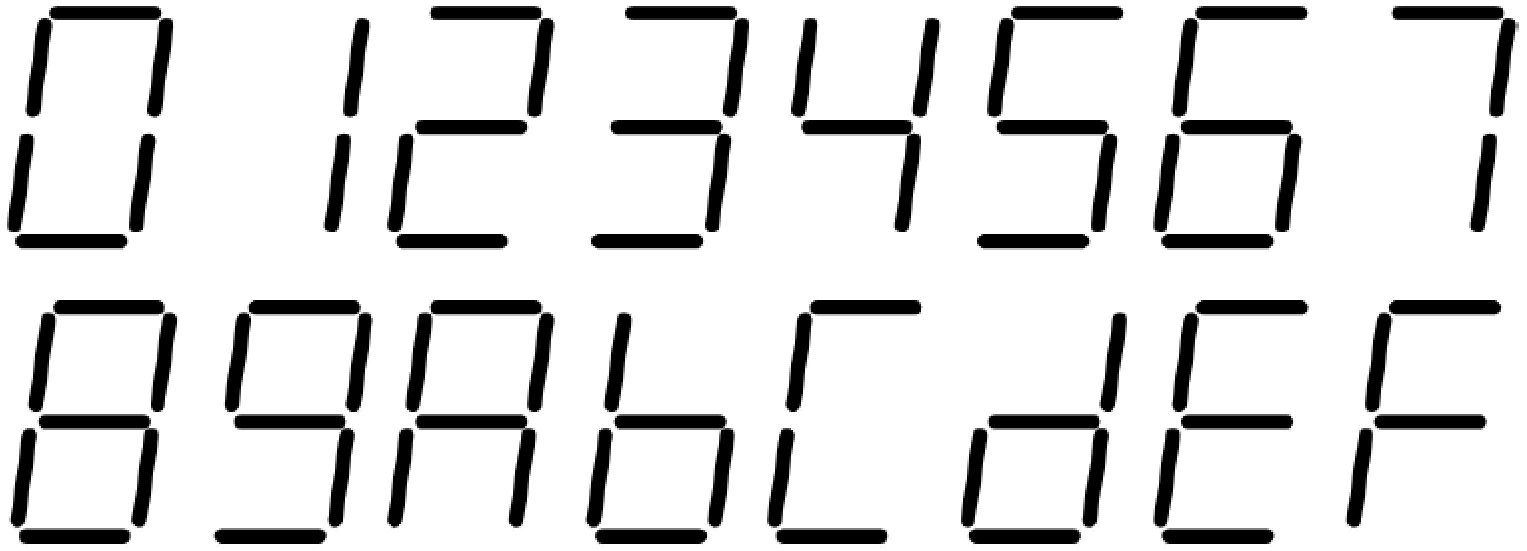
5

```verilog
    //registered outputs
    always@(posedge clk or posedge reset) begin
        if(reset) begin
            {candy_out, change_out} <= 2'b00;
        end
        else if (current_state != 2'b00) begin
            {candy_out, change_out} <= {candy, change};
        end
        else begin
            {candy_out, change_out} <= {candy_out, change_out};
        end
    end

    //money display
    assign money_display = current_state;
endmodule
```

DESIGN EXAMPLE

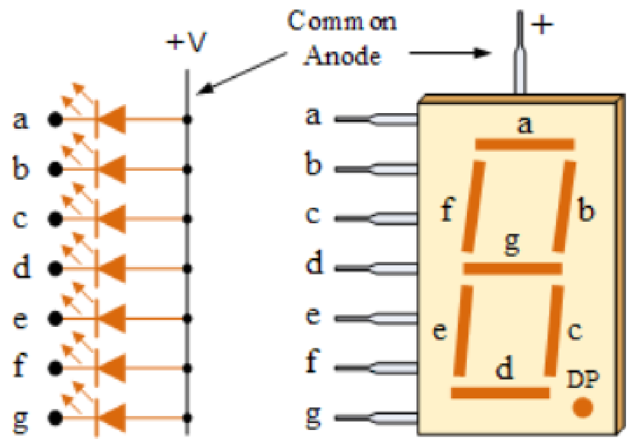
- Candy Vending Machine Project
  - 7-Segment Display Module
    - The following numbers can be displayed by setting the segments a, b, c, d, e, f, g, appropriately

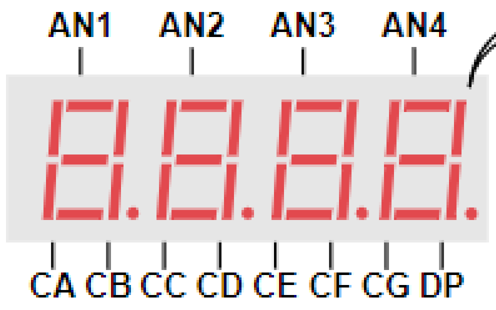




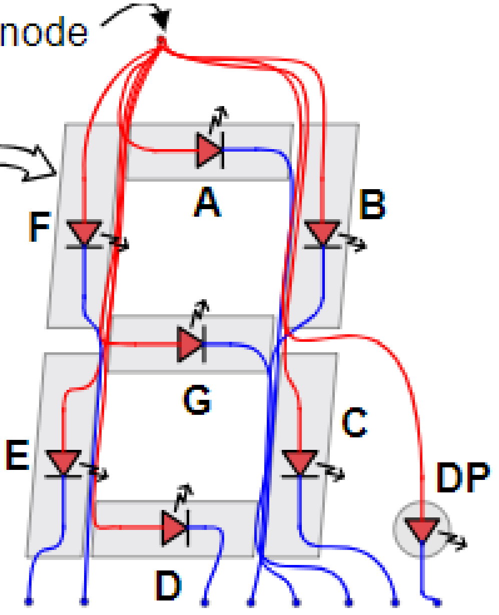
DESIGN EXAMPLE

- Candy Vending Machine Project
  - 7-Segment Display Module
    - The board uses a common anode display, which means that all of the anode connections for a given digit are tied together into a common circuit node
    - To illuminate a given segment in a given digit, a '1' must be applied to the digit's anode, and '0' must be applied to the segment's cathodes





Four-digit Seven Segment Display

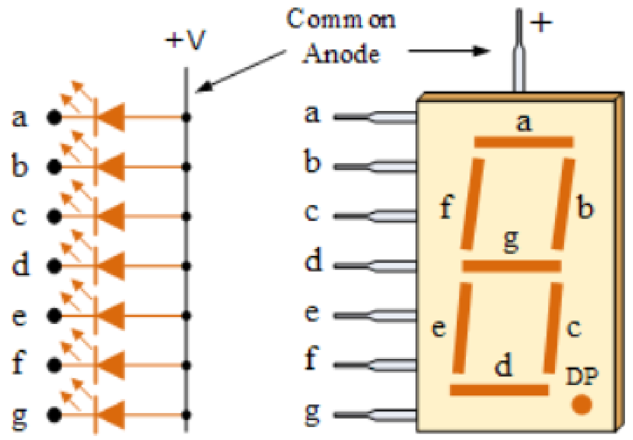


Individual cathodes

DESIGN EXAMPLE

- Candy Vending Machine Project
  - 7-Segment Display Module

	dp	g	f	e	d	c	b	a	binary	hexadecimal
0	1	1	0	0	0	0	0	0	0b11000000	0xc0
1	1	1	1	1	1	0	0	1	0b11111001	0xf9
2	1	0	1	0	0	1	0	0	0b10100100	0xa4
3	1	0	1	1	0	0	0	0	0b10110000	0xb0
4	1	0	0	1	1	0	0	1	0b10011001	0x99
5	1	0	0	1	0	0	1	0	0b10010010	0x92
6	1	0	0	0	0	0	1	0	0b10000010	0x82
7	1	1	1	1	1	0	0	0	0b11111000	0xf8
8	1	0	0	0	0	0	0	0	0b10000000	0x80
9	1	0	0	1	0	0	0	0	0b10010000	0x90
A	1	0	0	0	1	0	0	0	0b10001000	0x88
b	1	0	0	0	0	0	1	1	0b10000011	0x83
C	1	1	0	0	0	1	1	0	0b11000110	0xc6
d	1	0	1	0	0	0	0	1	0b10100001	0xa1
E	1	0	0	0	0	1	1	0	0b10000110	0x86
F	1	0	0	0	1	1	1	0	0b10001110	0x8e
OFF	1	1	1	1	1	1	1	1	0b11111111	0xff



DESIGN EXAMPLE

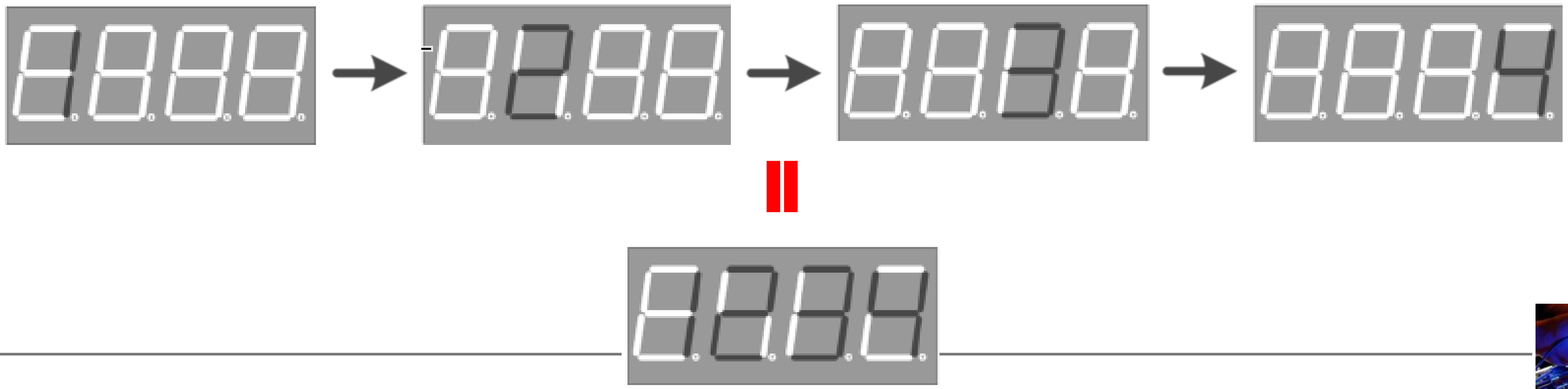
- Candy Vending Machine Project
  - 7-Segment Display Module
    - Controlling 4 common terminals is termed scanning method
      - Reduces number of utilized I/O of a device
      - Common data line for all the 4 segment
      - Select lines for each 7-Segment in the array

    - If the control inputs a value of '1' to the common terminal of the 7-segment A number or letter is displayed on the selected Segment

DESIGN EXAMPLE

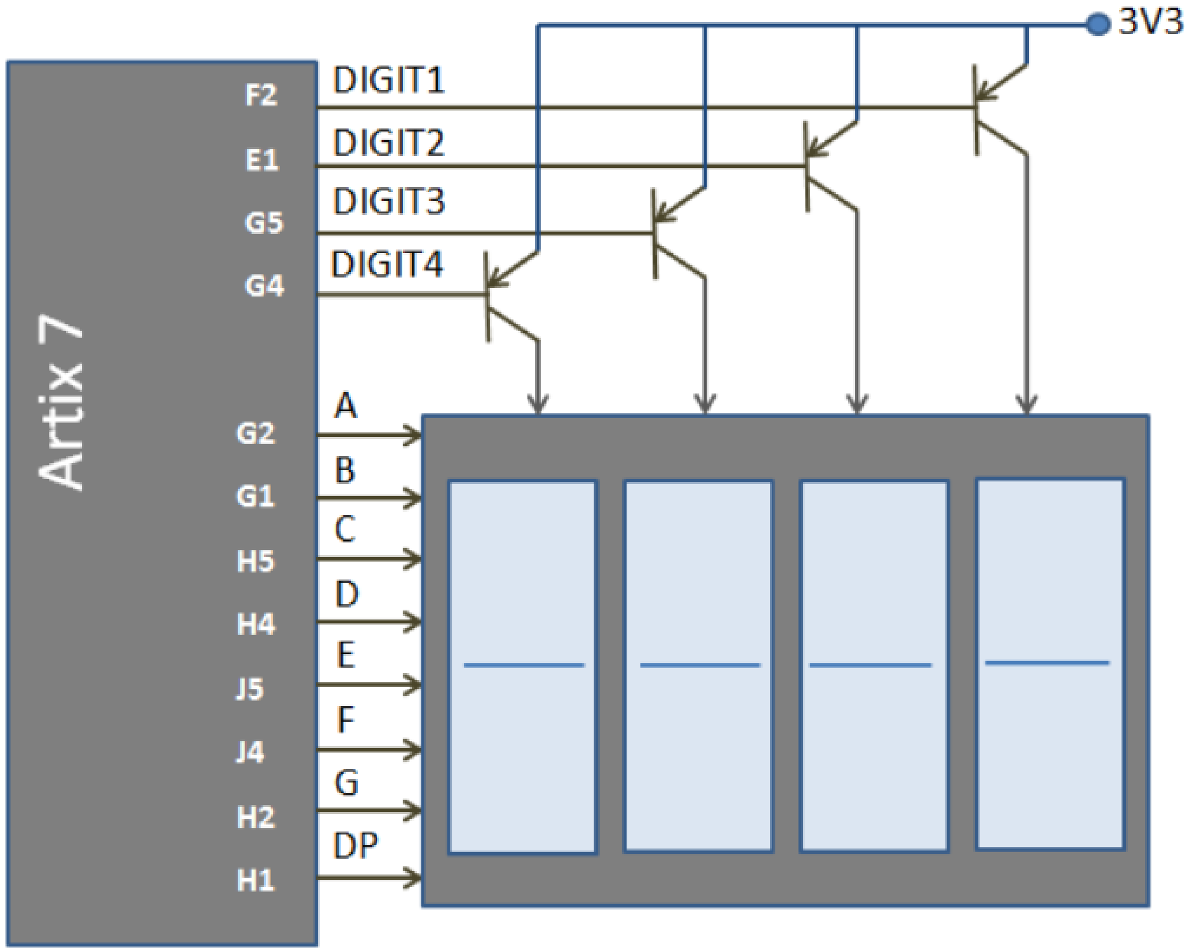
❖ Candy Vending Machine Project

- 7-Segment Display Module
  - To display the number "1234" on four 7-Segments, pass value '1' to the first common terminal and '0' to the remainder of the common terminal to select the first
    - Data to be displayed (1) is passed to seg_data
  - The second segment is selected with a '1' that common terminal and '0'to the rest
    - The data to be displayed (2) is passed to seg_data.
    - This is repeated for the 3rd and 4th digits



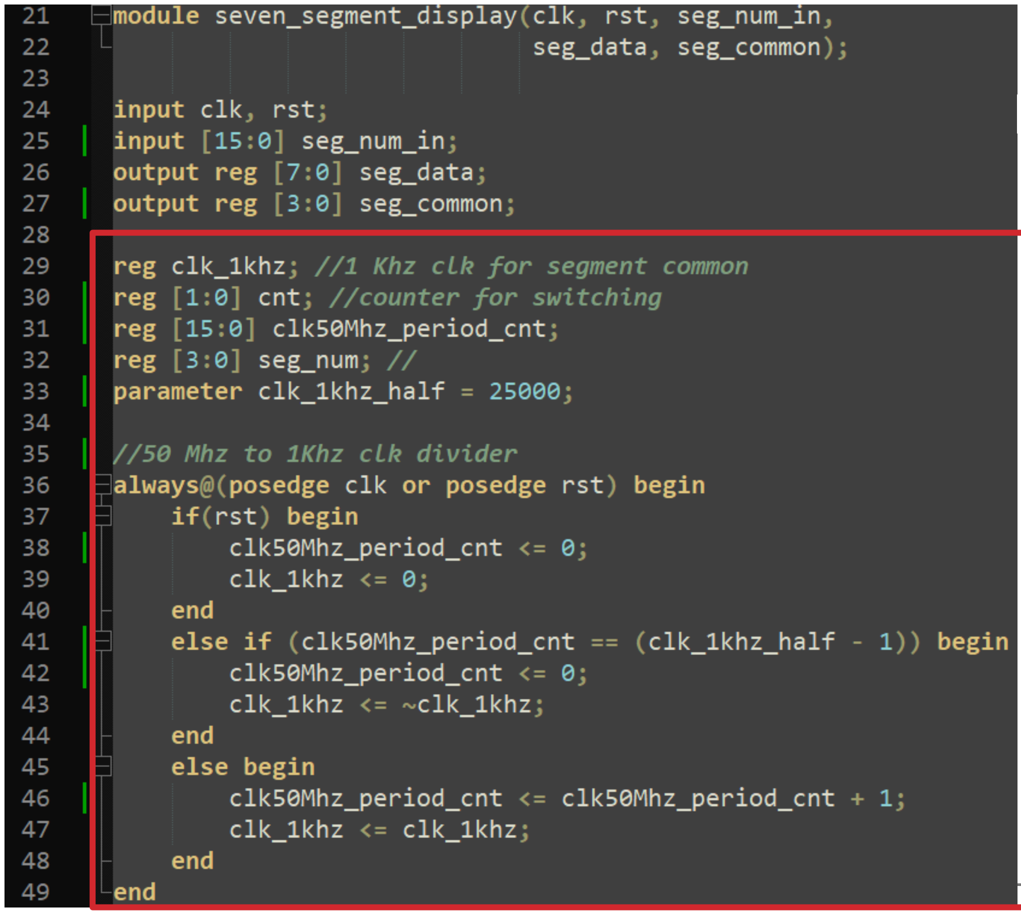
DESIGN EXAMPLE

- Candy Vending Machine Project
  - 7-Segment Display Module (Schematic)

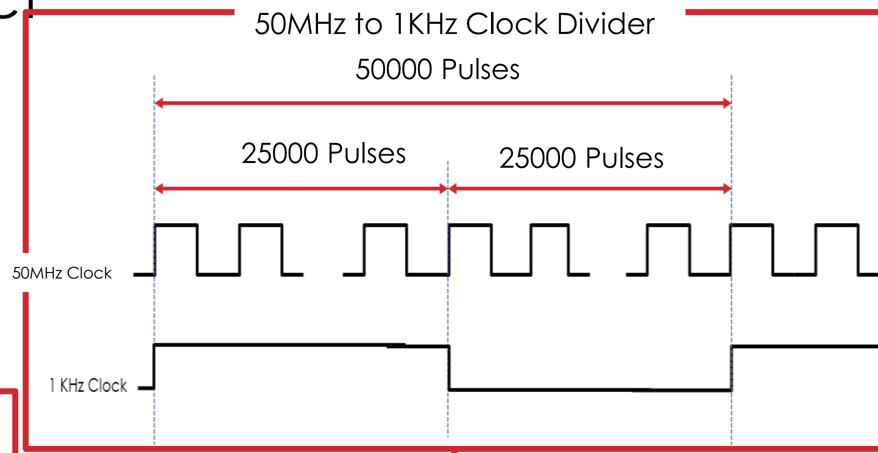


DESIGN EXAMPLE

- Candy Vending Machine Project
  - 7-Segment Display Verilog Code

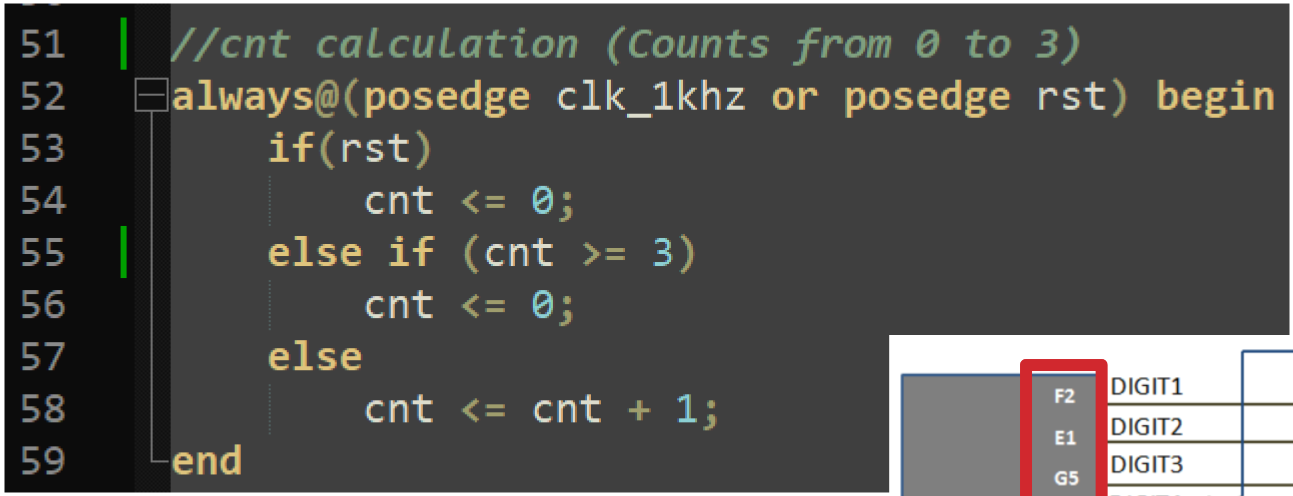


```
module seven_segment_display(clk, rst, seg_num_in,
                             seg_data, seg_common);

input clk, rst;
input [15:0] seg_num_in;
output reg [7:0] seg_data;
output reg [3:0] seg_common;

reg clk_1khz; //1 Khz clk for segment common
reg [1:0] cnt; //counter for switching
reg [15:0] clk50Mhz_period_cnt;
reg [3:0] seg_num; //
parameter clk_1khz_half = 25000;

//50 Mhz to 1Khz clk divider
always@(posedge clk or posedge rst) begin
    if(rst) begin
        clk50Mhz_period_cnt <= 0;
        clk_1khz <= 0;
    end
    else if (clk50Mhz_period_cnt == (clk_1khz_half - 1)) begin
        clk50Mhz_period_cnt <= 0;
        clk_1khz <= ~clk_1khz;
    end
    else begin
        clk50Mhz_period_cnt <= clk50Mhz_period_cnt + 1;
        clk_1khz <= clk_1khz;
    end
end
```



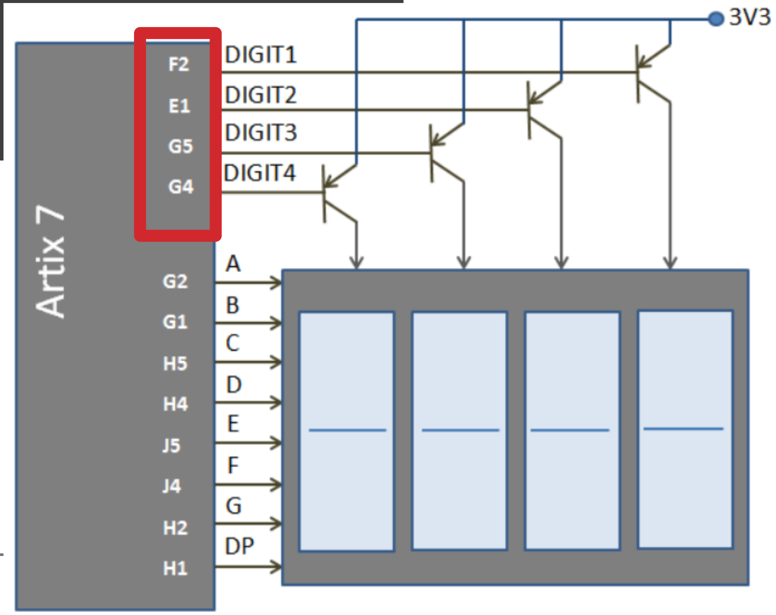
DESIGN EXAMPLE

- Candy Vending Machine Project
  - 7-Segment Display Verilog Code



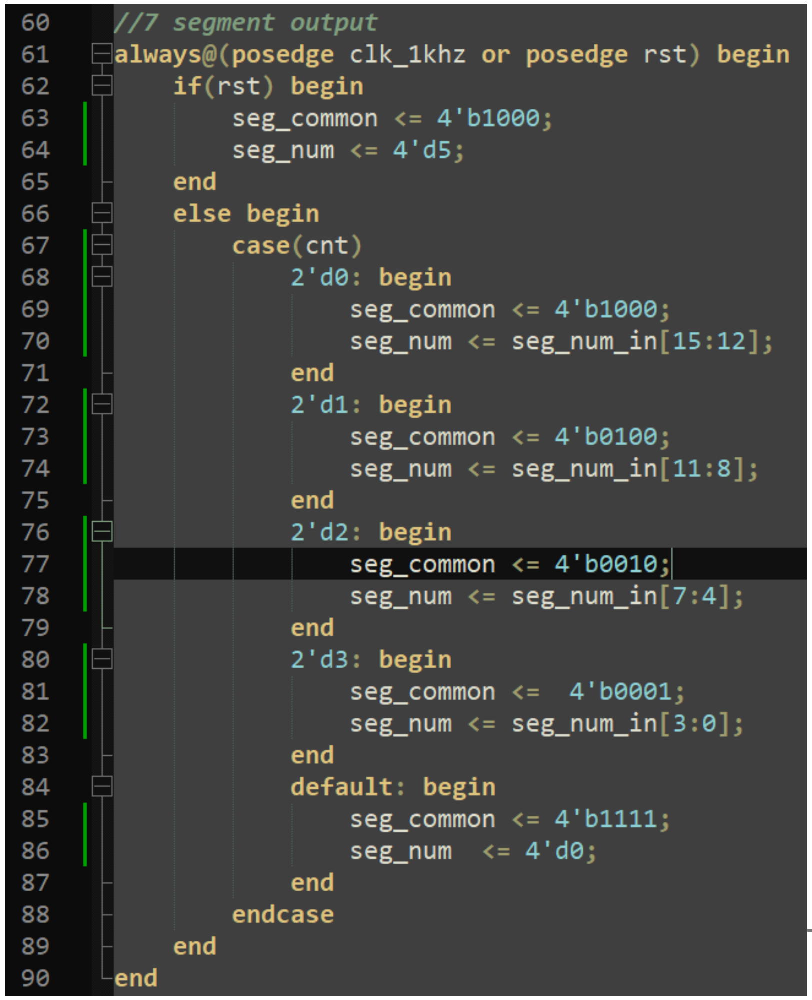
```
//cnt calculation (Counts from 0 to 3)
always@(posedge clk_1khz or posedge rst) begin
    if(rst)
        cnt <= 0;
    else if (cnt >= 3)
        cnt <= 0;
    else
        cnt <= cnt + 1;
end
```

Count from 0 to 3



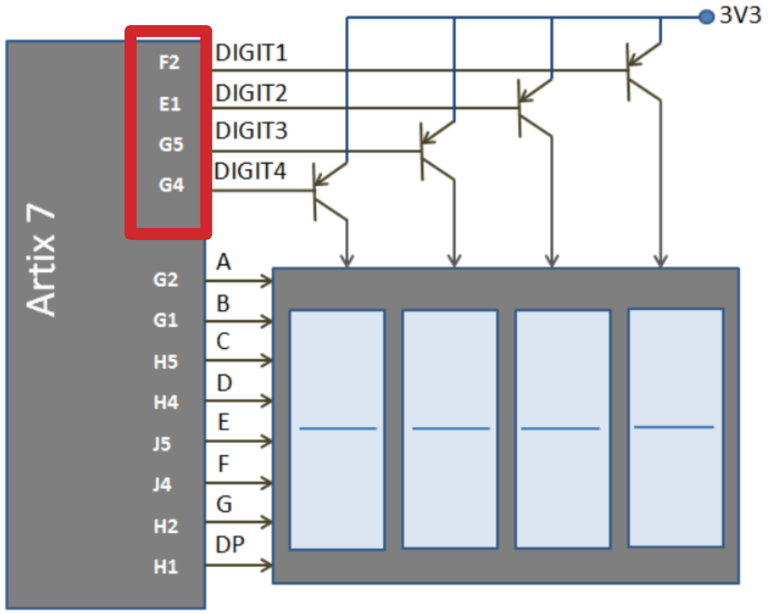
DESIGN EXAMPLE

- Candy Vending Machine Project
  - 7-Segment Display Verilog Code



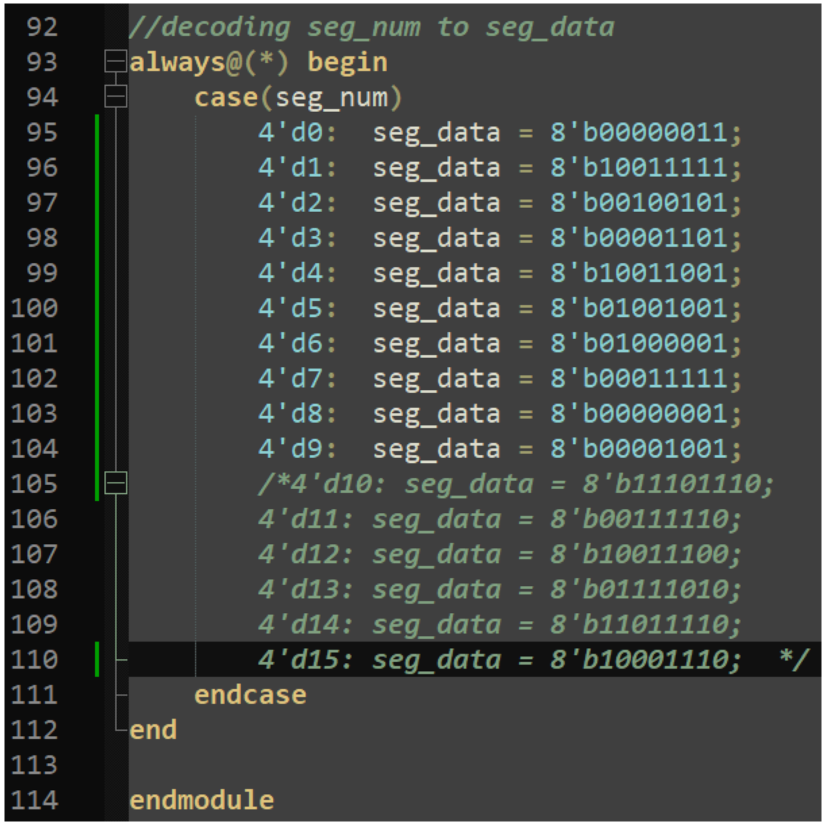
```verilog
//7 segment output
always@(posedge clk_1khz or posedge rst) begin
    if(rst) begin
        seg_common <= 4'b1000;
        seg_num <= 4'd5;
    end
    else begin
        case(cnt)
            2'd0: begin
                seg_common <= 4'b1000;
                seg_num <= seg_num_in[15:12];
            end
            2'd1: begin
                seg_common <= 4'b0100;
                seg_num <= seg_num_in[11:8];
            end
            2'd2: begin
                seg_common <= 4'b0010;
                seg_num <= seg_num_in[7:4];
            end
            2'd3: begin
                seg_common <=  4'b0001;
                seg_num <= seg_num_in[3:0];
            end
            default: begin
                seg_common <= 4'b1111;
                seg_num  <= 4'd0;
            end
        endcase
    end
end
```

Count from 0 to 3

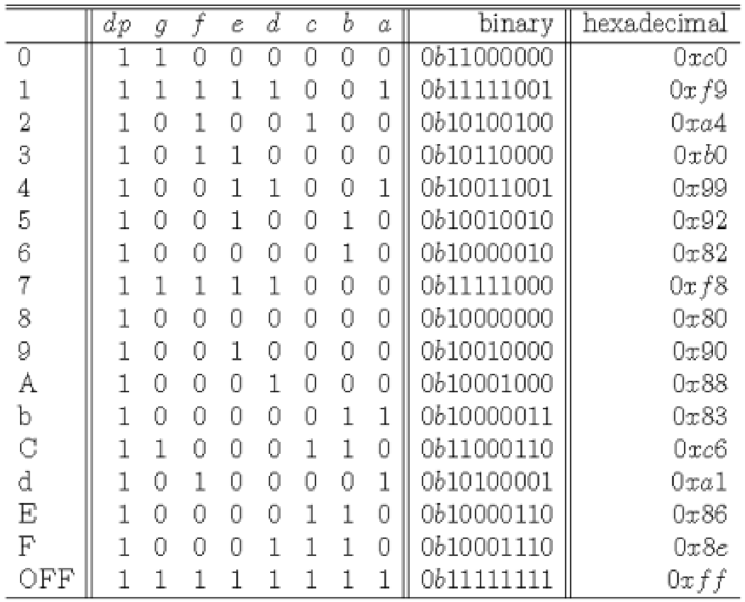


DESIGN EXAMPLE

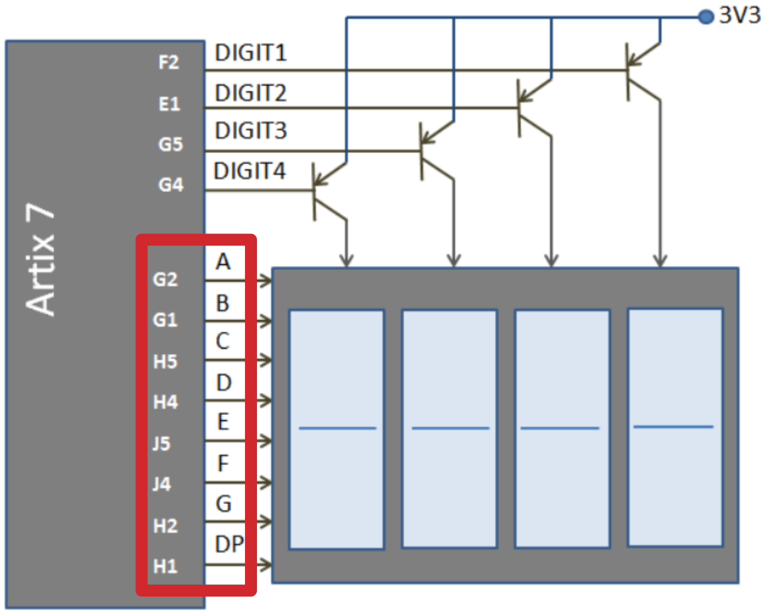
- Candy Vending Machine Project
  - 7-Segment Display Verilog Code

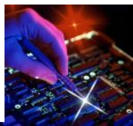


```
//decoding seg_num to seg_data
always@(*) begin
    case(seg_num)
        4'd0:  seg_data = 8'b00000011;
        4'd1:  seg_data = 8'b10011111;
        4'd2:  seg_data = 8'b00100101;
        4'd3:  seg_data = 8'b00001101;
        4'd4:  seg_data = 8'b10011001;
        4'd5:  seg_data = 8'b01001001;
        4'd6:  seg_data = 8'b01000001;
        4'd7:  seg_data = 8'b00011111;
        4'd8:  seg_data = 8'b00000001;
        4'd9:  seg_data = 8'b00001001;
        /*4'd10: seg_data = 8'b11101110;
        4'd11: seg_data = 8'b00111110;
        4'd12: seg_data = 8'b10011100;
        4'd13: seg_data = 8'b01111010;
        4'd14: seg_data = 8'b11011110;
        4'd15: seg_data = 8'b10001110;   */
    endcase
end

endmodule
```



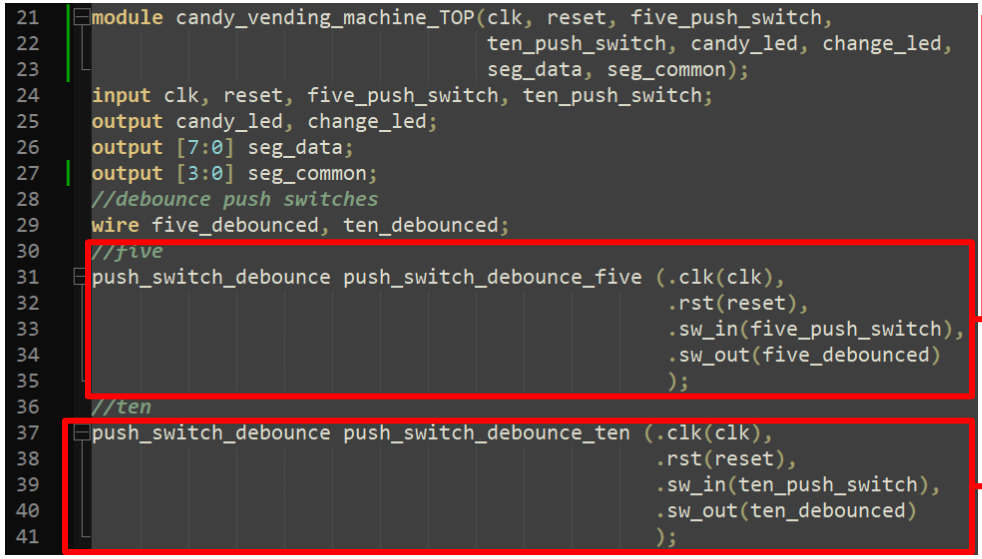
	dp	g	f	e	d	c	b	a	binary	hexadecimal
0	1	1	0	0	0	0	0	0	0b11000000	$0xc0$
1	1	1	1	1	1	0	0	1	0b11111001	$0xf9$
2	1	0	1	0	0	1	0	0	0b10100100	$0xa4$
3	1	0	1	1	0	0	0	0	0b10110000	$0xb0$
4	1	0	0	1	1	0	0	1	0b10011001	$0x99$
5	1	0	0	1	0	0	1	0	0b10010010	$0x92$
6	1	0	0	0	0	0	1	0	0b10000010	$0x82$
7	1	1	1	1	1	0	0	0	0b11111000	$0xf8$
8	1	0	0	0	0	0	0	0	0b10000000	$0x80$
9	1	0	0	1	0	0	0	0	0b10010000	$0x90$
A	1	0	0	0	1	0	0	0	0b10001000	$0x88$
b	1	0	0	0	0	0	1	1	0b10000011	$0x83$
C	1	1	0	0	0	1	1	0	0b11000110	$0xc6$
d	1	0	1	0	0	0	0	1	0b10100001	$0xa1$
E	1	0	0	0	0	1	1	0	0b10000110	$0x86$
F	1	0	0	0	1	1	1	0	0b10001110	$0x8e$
OFF	1	1	1	1	1	1	1	1	0b11111111	$0xff$



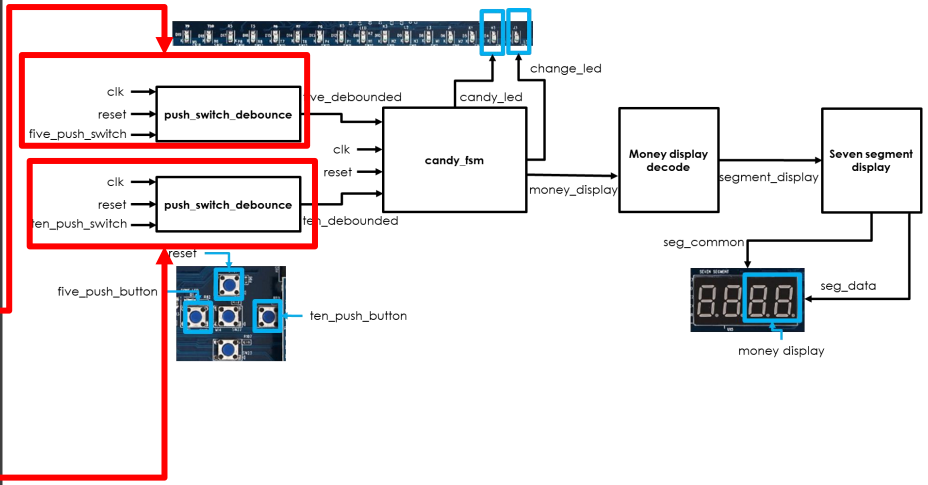


DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy Vending Machine Top Module Verilog Code

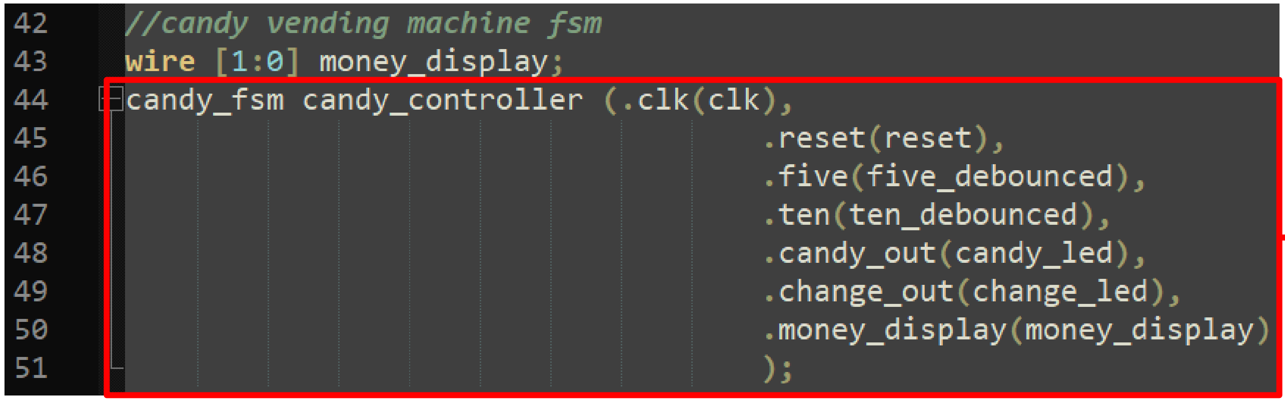


```verilog
module candy_vending_machine_TOP(clk, reset, five_push_switch,
                                  ten_push_switch, candy_led, change_led,
                                  seg_data, seg_common);
input clk, reset, five_push_switch, ten_push_switch;
output candy_led, change_led;
output [7:0] seg_data;
output [3:0] seg_common;
//debounce push switches
wire five_debounced, ten_debounced;
//five
push_switch_debounce push_switch_debounce_five (.clk(clk),
                                                .rst(reset),
                                                .sw_in(five_push_switch),
                                                .sw_out(five_debounced)
                                                );
//ten
push_switch_debounce push_switch_debounce_ten (.clk(clk),
                                               .rst(reset),
                                               .sw_in(ten_push_switch),
                                               .sw_out(ten_debounced)
                                               );
```

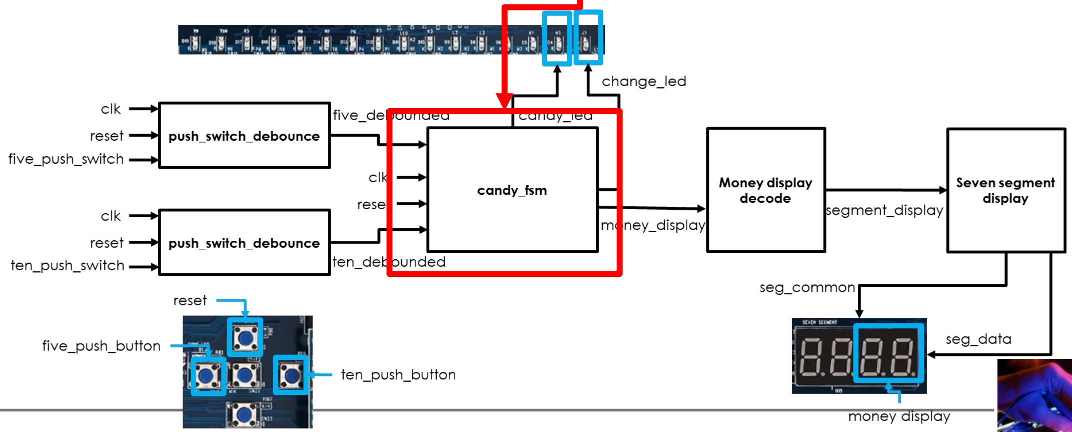


DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy Vending Machine Top Module Verilog Code

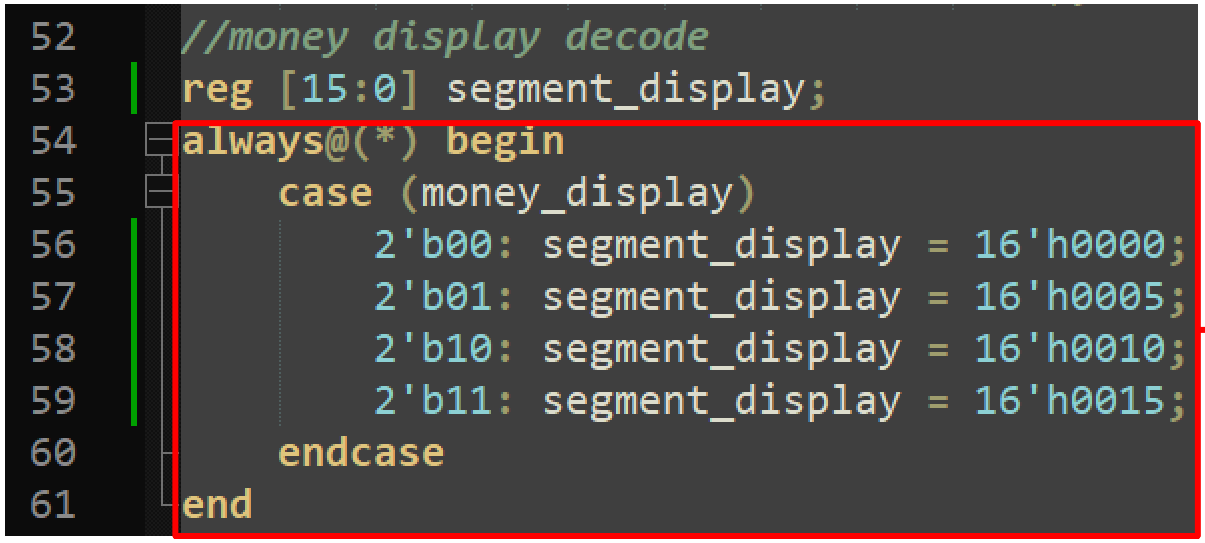


```
//candy vending machine fsm
wire [1:0] money_display;
candy_fsm candy_controller (.clk(clk),
                            .reset(reset),
                            .five(five_debounced),
                            .ten(ten_debounced),
                            .candy_out(candy_led),
                            .change_out(change_led),
                            .money_display(money_display)
                            );
```

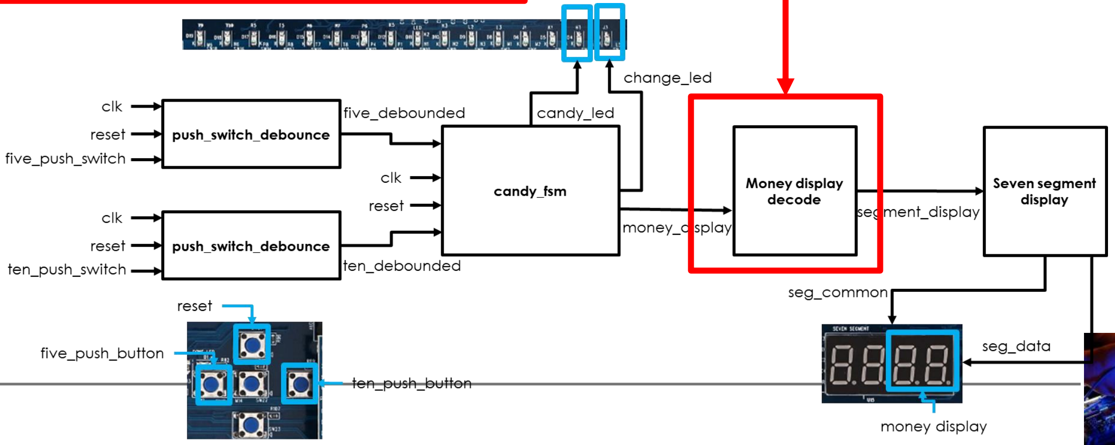


DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy Vending Machine Top Module Verilog Code

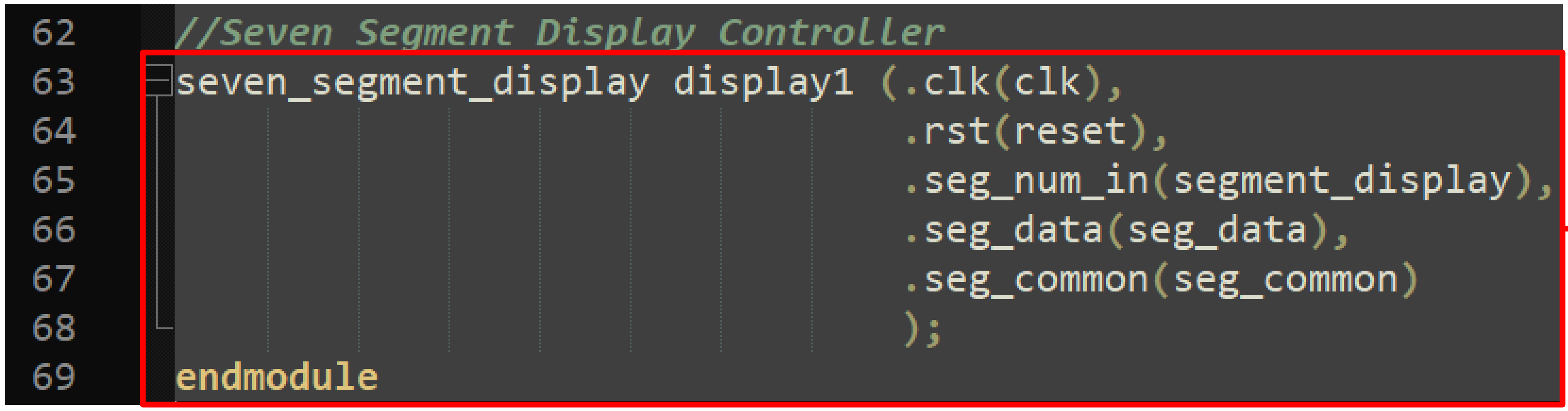


```
//money display decode
reg [15:0] segment_display;
always@(*) begin
    case (money_display)
        2'b00: segment_display = 16'h0000;
        2'b01: segment_display = 16'h0005;
        2'b10: segment_display = 16'h0010;
        2'b11: segment_display = 16'h0015;
    endcase
end
```

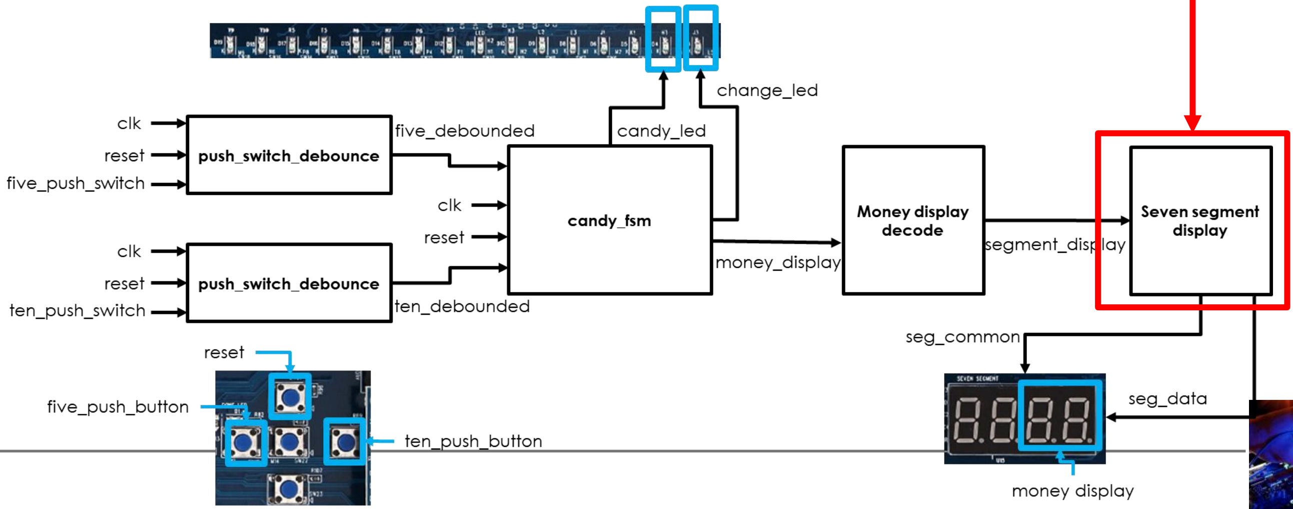


DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy Vending Machine Top Module Verilog Code

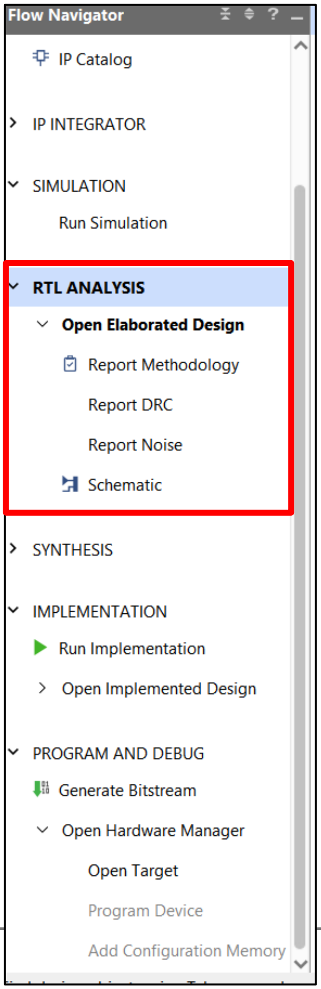


```verilog
//Seven Segment Display Controller
seven_segment_display display1 (.clk(clk),
                                .rst(reset),
                                .seg_num_in(segment_display),
                                .seg_data(seg_data),
                                .seg_common(seg_common)
                                );
endmodule
```



DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy Vending Machine Top Module Elaborated Schematic



Flow Navigator

IP Catalog

IP INTEGRATOR

SIMULATION

Run Simulation

RTL ANALYSIS

Open Elaborated Design

Report Methodology

Report DRC

Report Noise

Schematic

SYNTHESIS

IMPLEMENTATION

Run Implementation

Open Implemented Design

PROGRAM AND DEBUG

Generate Bitstream

Open Hardware Manager

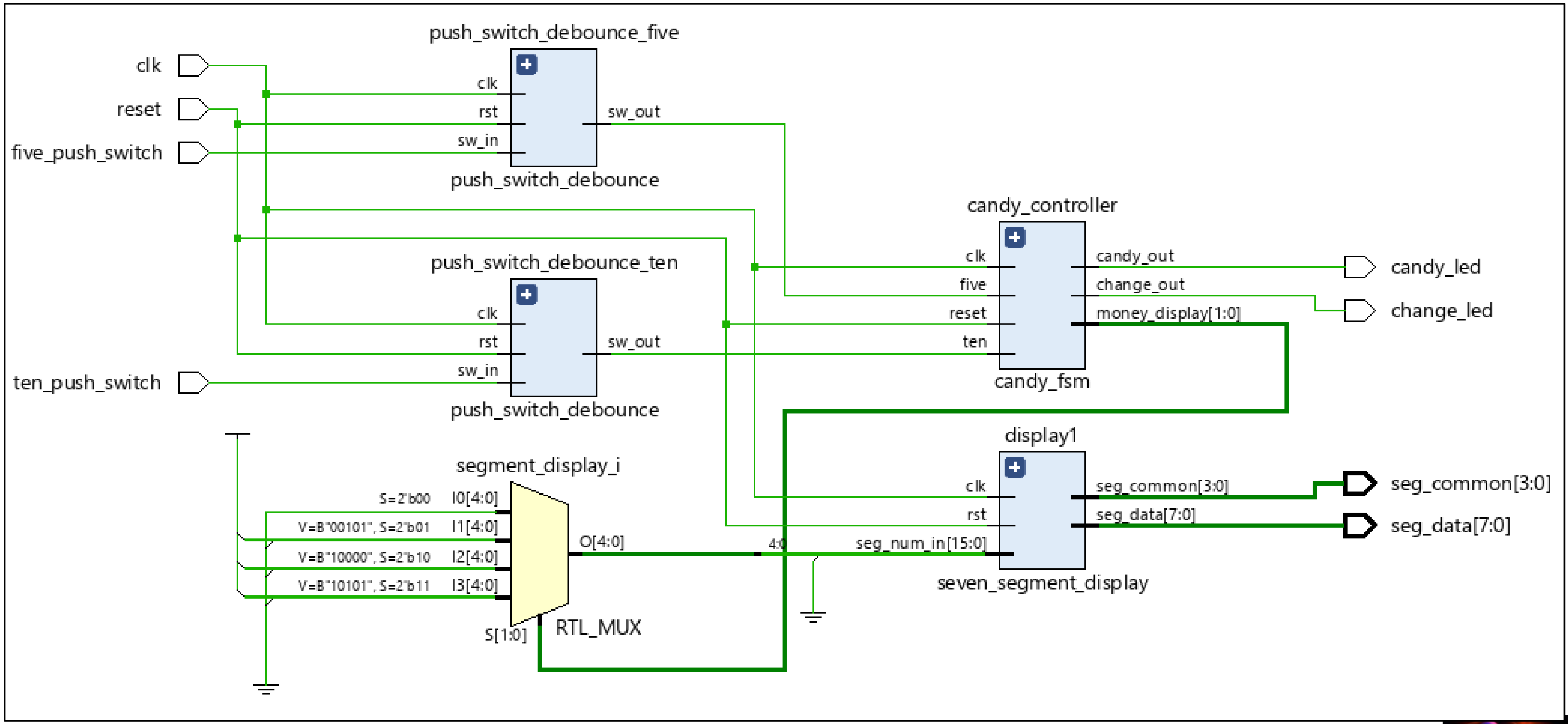
Open Target

Program Device

Add Configuration Memory

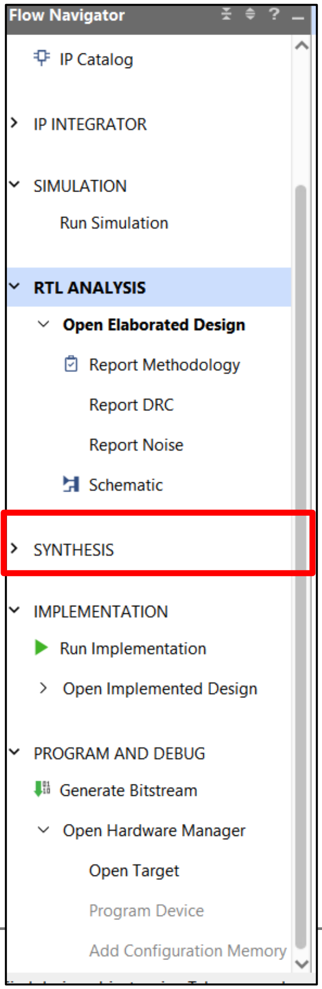
DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy Vending Machine Top Module Elaborated Schematic



DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy Vending Machine Top Module Synthesis Schematic



Flow Navigator

IP Catalog

IP INTEGRATOR

SIMULATION

Run Simulation

RTL ANALYSIS

Open Elaborated Design

Report Methodology

Report DRC

Report Noise

Schematic

SYNTHESIS

IMPLEMENTATION

Run Implementation

Open Implemented Design

PROGRAM AND DEBUG

Generate Bitstream

Open Hardware Manager

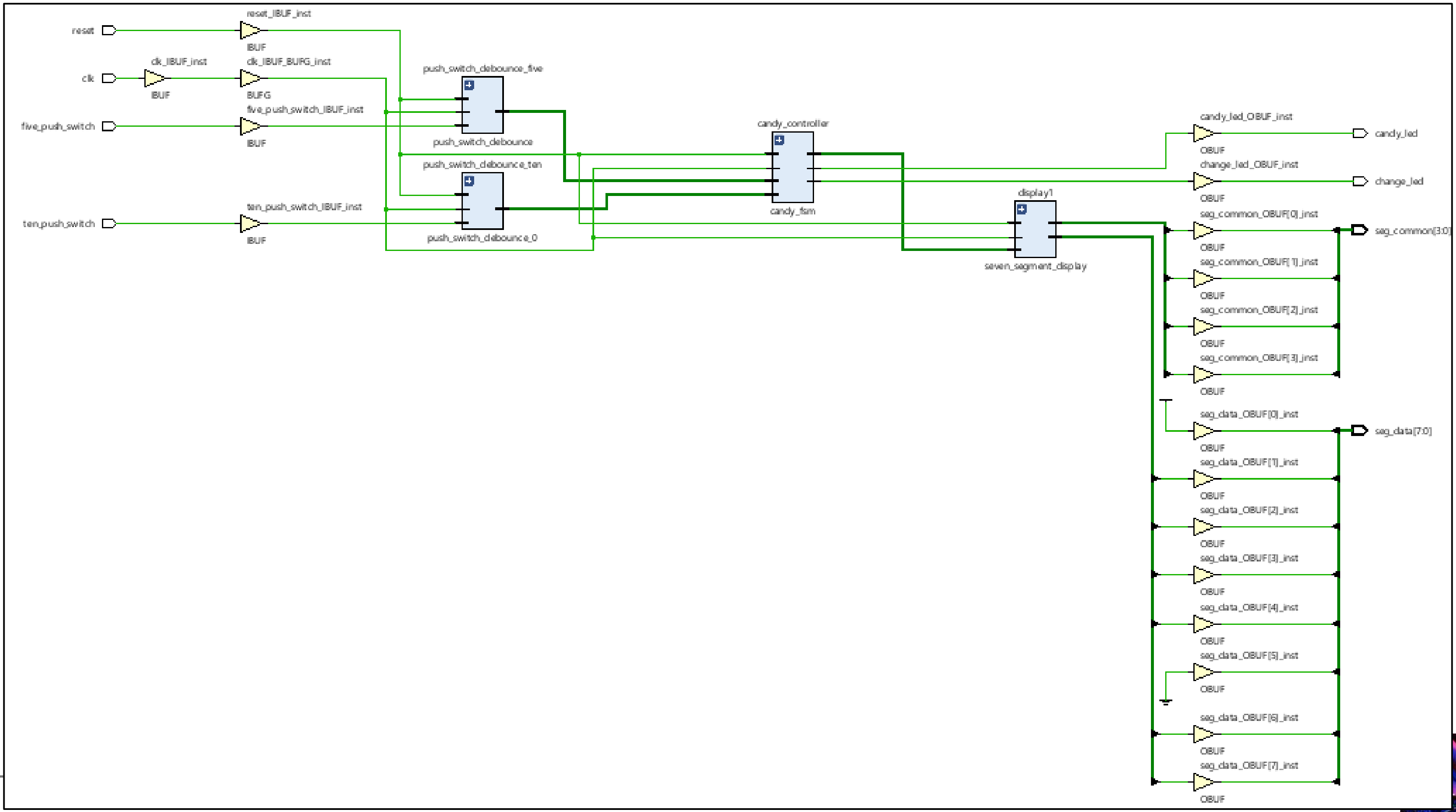
Open Target

Program Device

Add Configuration Memory

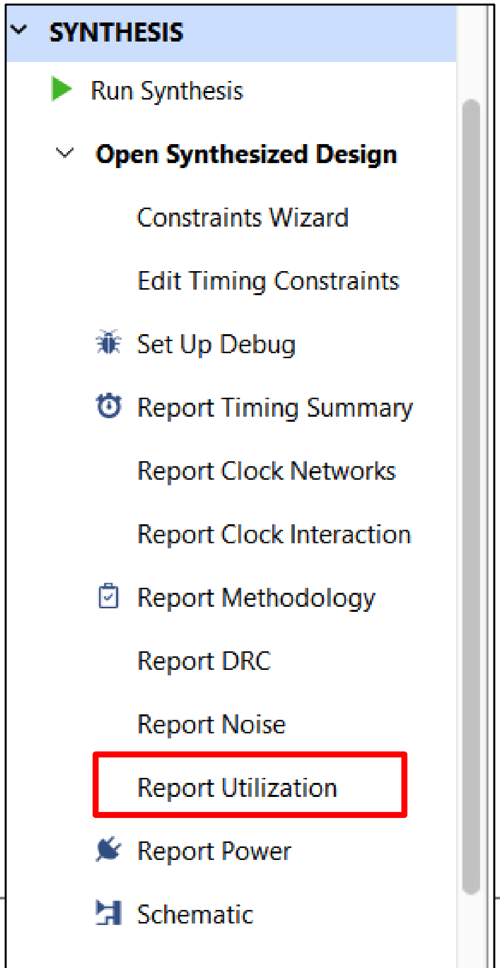
DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy Vending Machine Top Module Synthesis Schematic



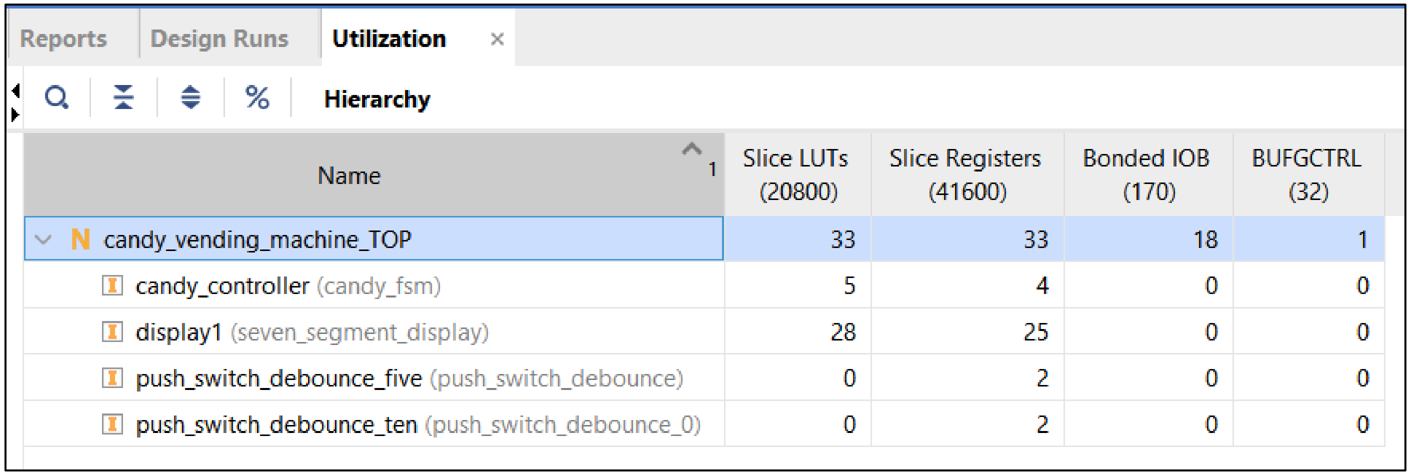
DESIGN EXAMPLE

- Candy Vending Machine Project
  - Candy Vending Machine Top Module Synthesis Results



DESIGN EXAMPLE

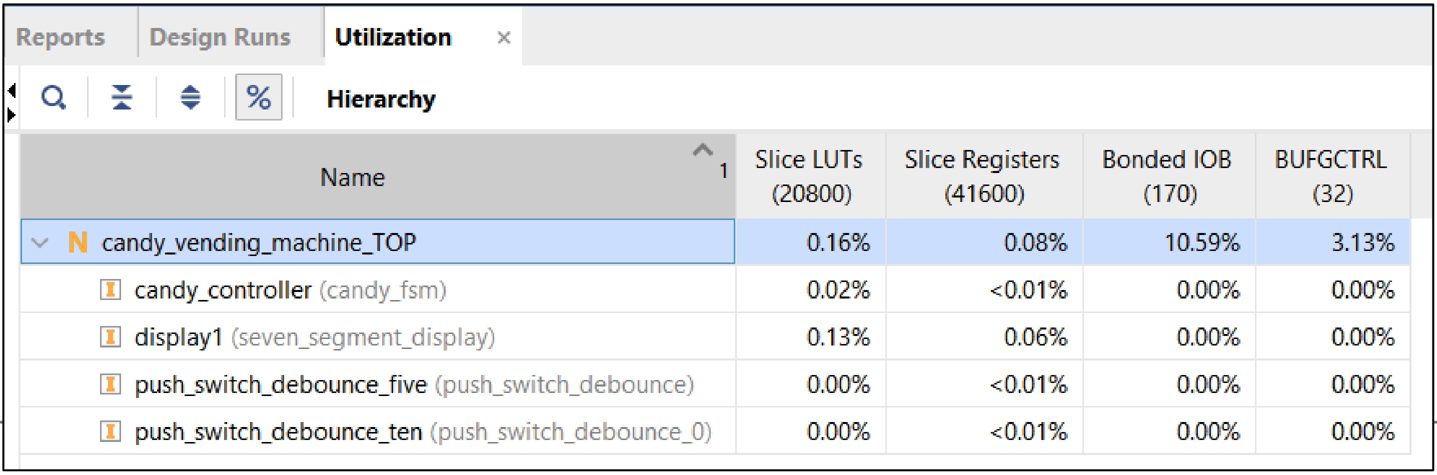
- Candy Vending Machine Project
  - Candy Vending Machine Top Module Synthesis Results



Reports | Design Runs | Utilization ×

Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (170)	BUFGCTRL (32)
candy_vending_machine_TOP	33	33	18	1
candy_controller (candy_fsm)	5	4	0	0
display1 (seven_segment_display)	28	25	0	0
push_switch_debounce_five (push_switch_debounce)	0	2	0	0
push_switch_debounce_ten (push_switch_debounce_0)	0	2	0	0



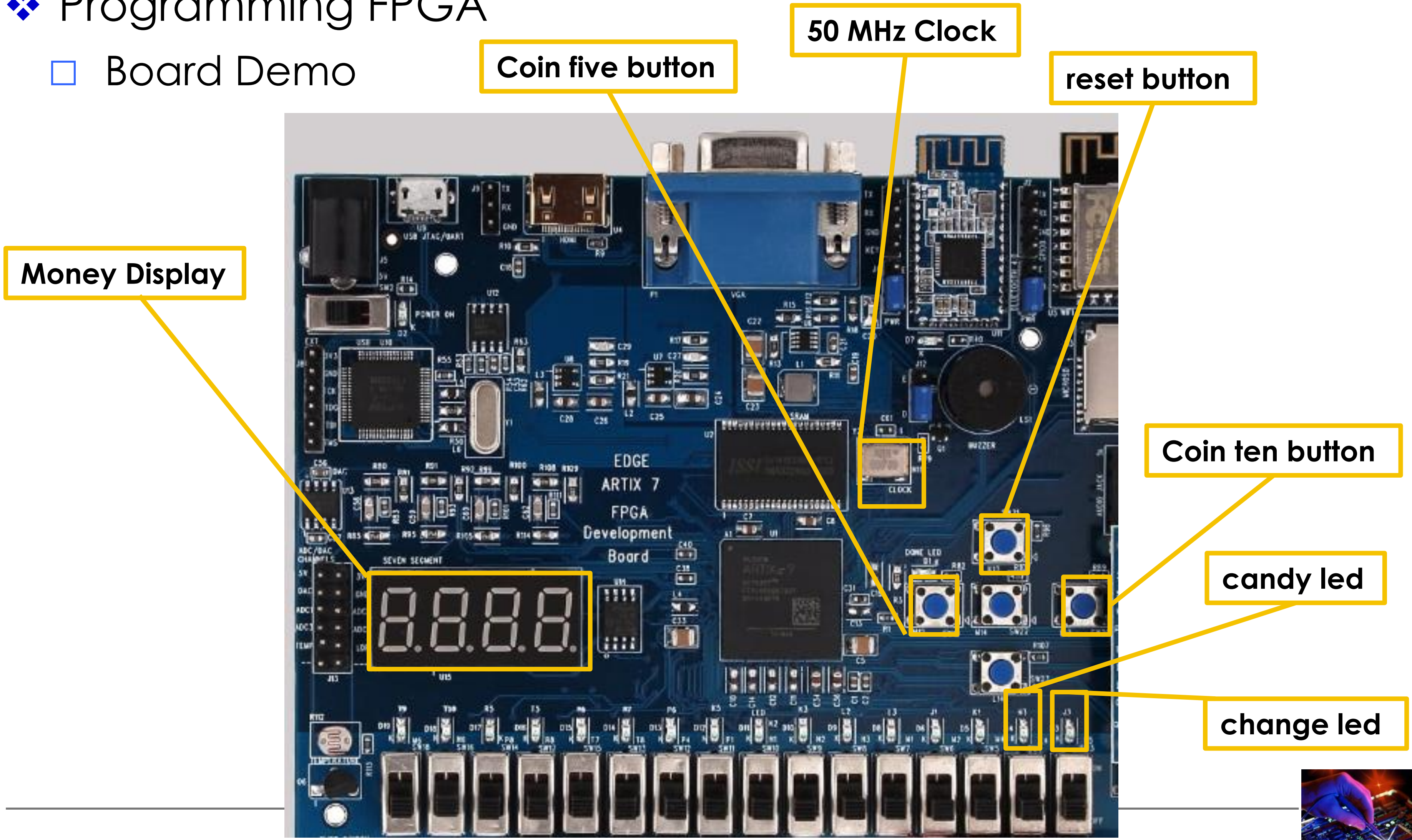
Reports | Design Runs | Utilization ×

Hierarchy

Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (170)	BUFGCTRL (32)
candy_vending_machine_TOP	0.16%	0.08%	10.59%	3.13%
candy_controller (candy_fsm)	0.02%	<0.01%	0.00%	0.00%
display1 (seven_segment_display)	0.13%	0.06%	0.00%	0.00%
push_switch_debounce_five (push_switch_debounce)	0.00%	<0.01%	0.00%	0.00%
push_switch_debounce_ten (push_switch_debounce_0)	0.00%	<0.01%	0.00%	0.00%

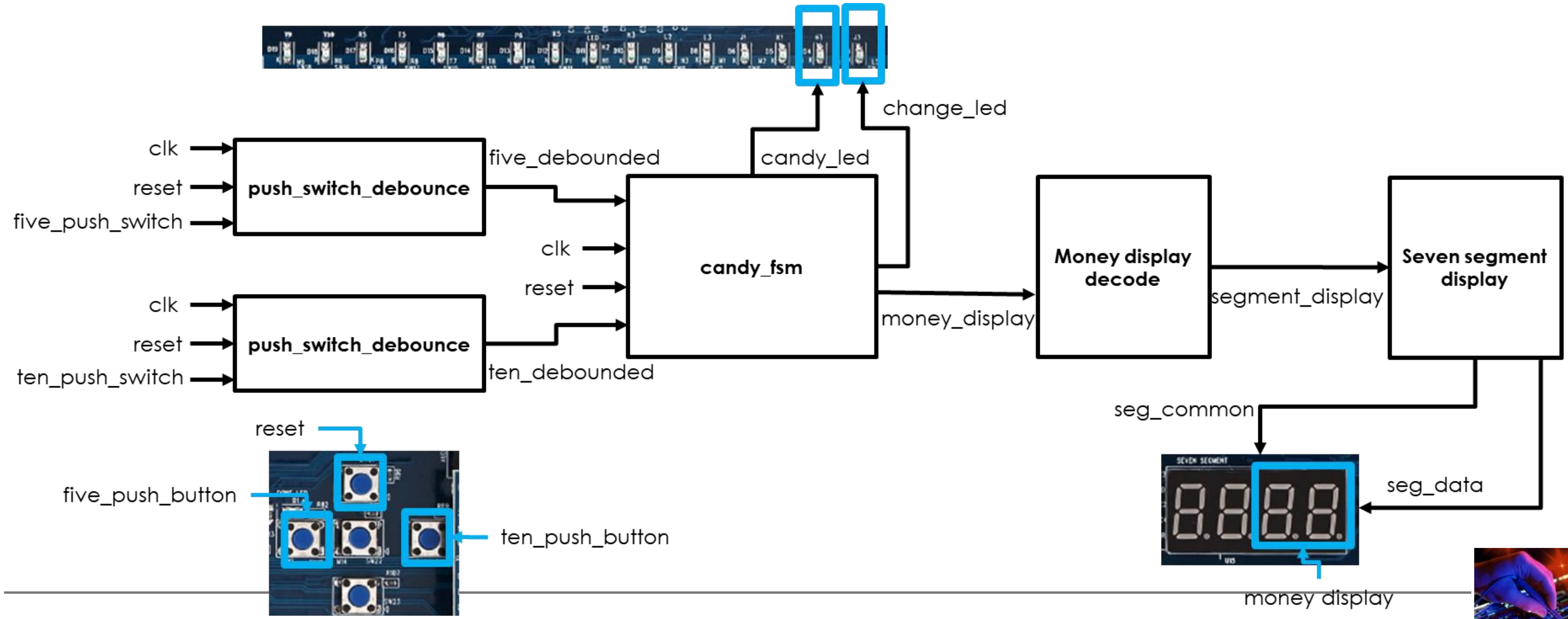
DESIGN EXAMPLE

- Programming FPGA
  - Board Demo



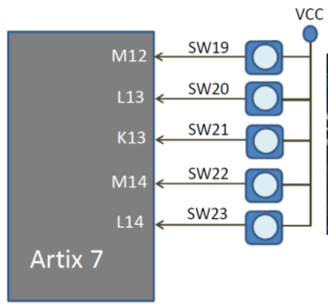
DESIGN EXAMPLE

- Implement Candy Vending Machine on Edge Artix 7 FPGA Board
  - Input: clk, reset, five_push_switch, ten_push_switch
  - Output: candy_led, change_led
  - Output: [7:0] seg_data
  - Output: [7:0] seg_common

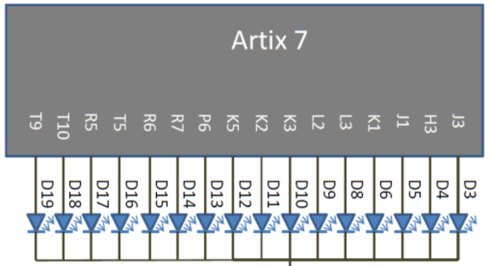


DESIGN EXAMPLE

- Vivado project
  - FPGA programming (pin mapping)



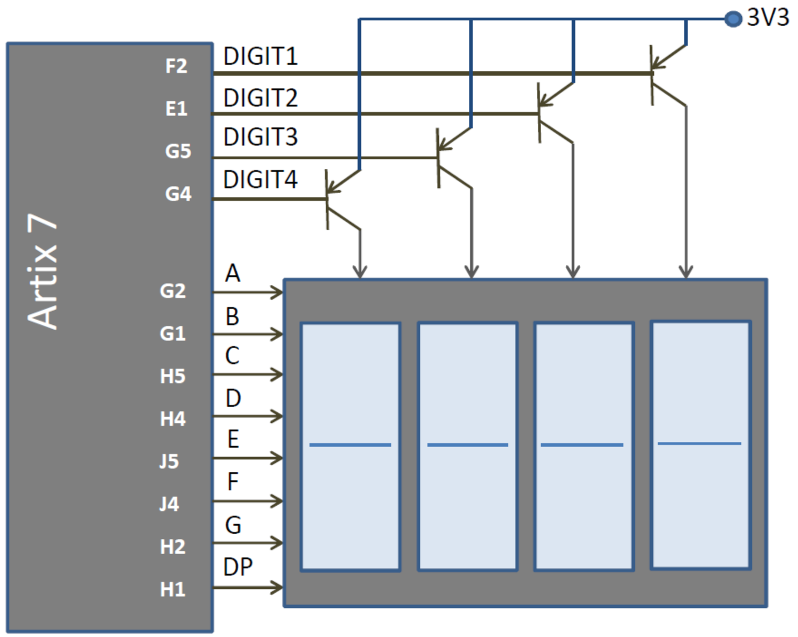
```
# Push Button
set_property -dict {PACKAGE_PIN K13 IOSTANDARD LVCMOS33 PULLDOWN true} [get_ports {pb[0]}]; #Button-top
set_property -dict {PACKAGE_PIN L14 IOSTANDARD LVCMOS33 PULLDOWN true} [get_ports {pb[1]}]; #Button-bottom
set_property -dict {PACKAGE_PIN M12 IOSTANDARD LVCMOS33 PULLDOWN true} [get_ports {pb[2]}]; #Button-left
set_property -dict {PACKAGE_PIN L13 IOSTANDARD LVCMOS33 PULLDOWN true} [get_ports {pb[3]}]; #Button-right
set_property -dict {PACKAGE_PIN M14 IOSTANDARD LVCMOS33 PULLDOWN true} [get_ports {pb[4]}]; #Button-center
```

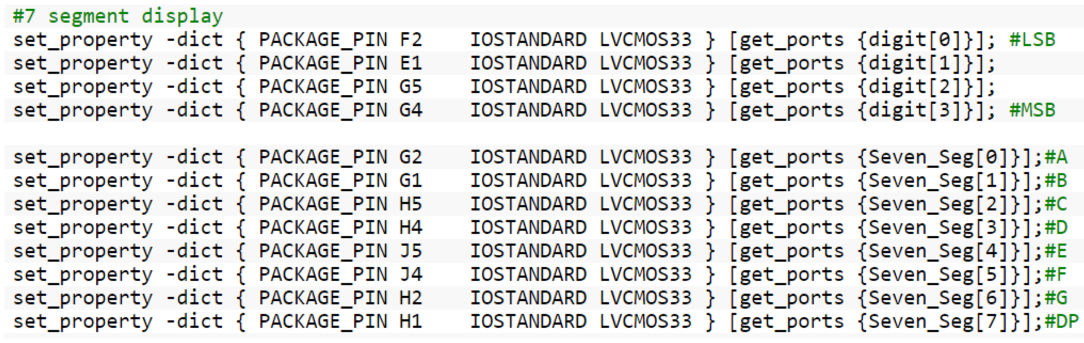


```
# LEDs
set_property -dict { PACKAGE_PIN J3    IOSTANDARD LVCMOS33 } [get_ports { led[0] }];#LSB
set_property -dict { PACKAGE_PIN H3    IOSTANDARD LVCMOS33 } [get_ports { led[1] }];
set_property -dict { PACKAGE_PIN J1    IOSTANDARD LVCMOS33 } [get_ports { led[2] }];
set_property -dict { PACKAGE_PIN K1    IOSTANDARD LVCMOS33 } [get_ports { led[3] }];
set_property -dict { PACKAGE_PIN L3    IOSTANDARD LVCMOS33 } [get_ports { led[4] }];
set_property -dict { PACKAGE_PIN L2    IOSTANDARD LVCMOS33 } [get_ports { led[5] }];
set_property -dict { PACKAGE_PIN K3    IOSTANDARD LVCMOS33 } [get_ports { led[6] }];
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { led[7] }];
set_property -dict { PACKAGE_PIN K5    IOSTANDARD LVCMOS33 } [get_ports { led[8] }];
set_property -dict { PACKAGE_PIN P6    IOSTANDARD LVCMOS33 } [get_ports { led[9] }];
set_property -dict { PACKAGE_PIN R7    IOSTANDARD LVCMOS33 } [get_ports { led[10] }];
set_property -dict { PACKAGE_PIN R6    IOSTANDARD LVCMOS33 } [get_ports { led[11] }];
set_property -dict { PACKAGE_PIN T5    IOSTANDARD LVCMOS33 } [get_ports { led[12] }];
set_property -dict { PACKAGE_PIN R5    IOSTANDARD LVCMOS33 } [get_ports { led[13] }];
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { led[14] }];
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { led[15] }];#MSB
```

DESIGN EXAMPLE

- Vivado project
  - FPGA programming (pin mapping)

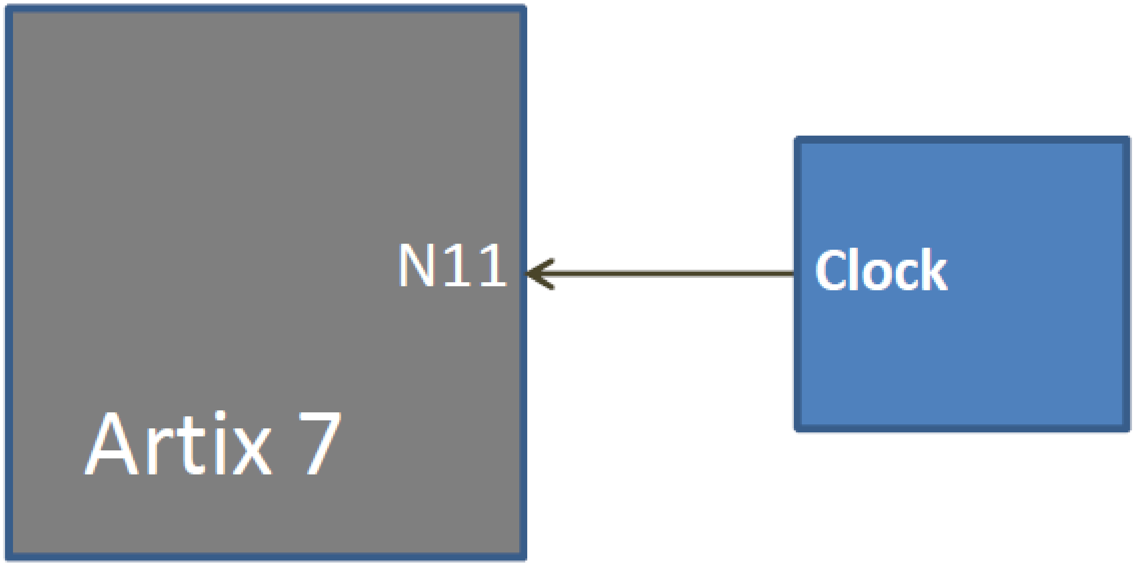




```
#7 segment display
set_property -dict { PACKAGE_PIN F2   IOSTANDARD LVCMOS33 } [get_ports {digit[0]}]; #LSB
set_property -dict { PACKAGE_PIN E1   IOSTANDARD LVCMOS33 } [get_ports {digit[1]}];
set_property -dict { PACKAGE_PIN G5   IOSTANDARD LVCMOS33 } [get_ports {digit[2]}];
set_property -dict { PACKAGE_PIN G4   IOSTANDARD LVCMOS33 } [get_ports {digit[3]}]; #MSB

set_property -dict { PACKAGE_PIN G2   IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[0]}];#A
set_property -dict { PACKAGE_PIN G1   IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[1]}];#B
set_property -dict { PACKAGE_PIN H5   IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[2]}];#C
set_property -dict { PACKAGE_PIN H4   IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[3]}];#D
set_property -dict { PACKAGE_PIN J5   IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[4]}];#E
set_property -dict { PACKAGE_PIN J4   IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[5]}];#F
set_property -dict { PACKAGE_PIN H2   IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[6]}];#G
set_property -dict { PACKAGE_PIN H1   IOSTANDARD LVCMOS33 } [get_ports {Seven_Seg[7]}];#DP
```

DESIGN EXAMPLE

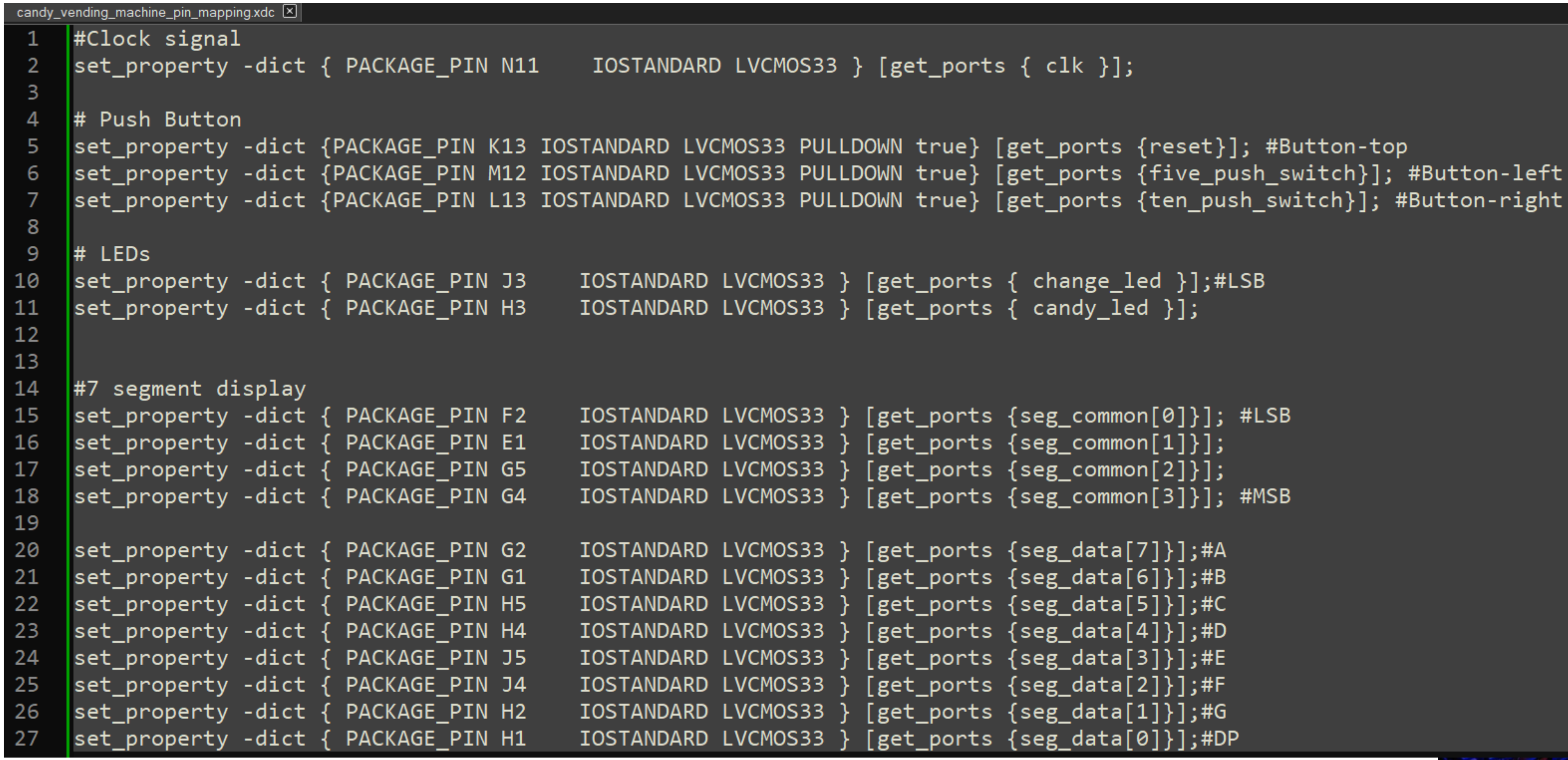
- Vivado project
  - FPGA programming (pin mapping)



```
# Clock signal
set_property -dict { PACKAGE_PIN N11    IOSTANDARD LVCMOS33 } [get_ports { clk }];
```

DESIGN EXAMPLE

- Vivado project
  - FPGA programming (mapping) – Create a constraint file
    - candy_vending_machine_pin_mapping.xdc

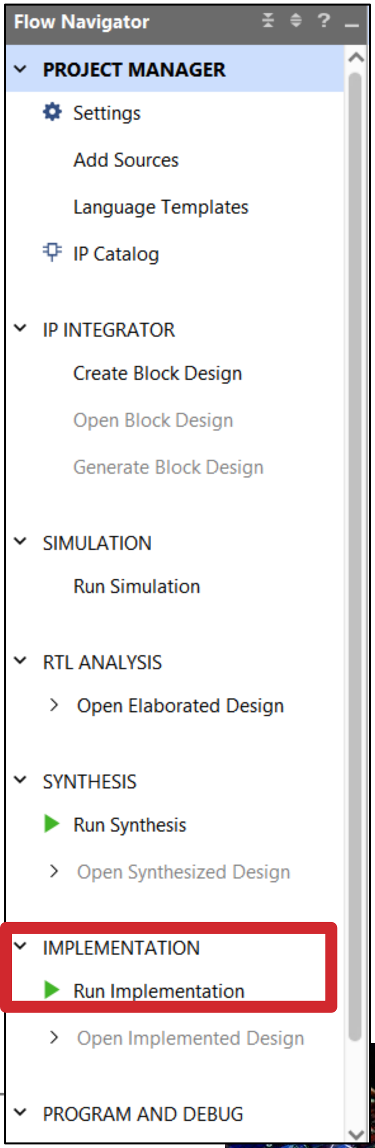


candy_vending_machine_pin_mapping.xdc

```
#Clock signal
set_property -dict { PACKAGE_PIN N11    IOSTANDARD LVCMOS33 } [get_ports { clk }];

# Push Button
set_property -dict {PACKAGE_PIN K13 IOSTANDARD LVCMOS33 PULLDOWN true} [get_ports {reset}]; #Button-top
set_property -dict {PACKAGE_PIN M12 IOSTANDARD LVCMOS33 PULLDOWN true} [get_ports {five_push_switch}]; #Button-left
set_property -dict {PACKAGE_PIN L13 IOSTANDARD LVCMOS33 PULLDOWN true} [get_ports {ten_push_switch}]; #Button-right

# LEDs
set_property -dict { PACKAGE_PIN J3    IOSTANDARD LVCMOS33 } [get_ports { change_led }];#LSB
set_property -dict { PACKAGE_PIN H3    IOSTANDARD LVCMOS33 } [get_ports { candy_led }];


#7 segment display
set_property -dict { PACKAGE_PIN F2    IOSTANDARD LVCMOS33 } [get_ports {seg_common[0]}]; #LSB
set_property -dict { PACKAGE_PIN E1    IOSTANDARD LVCMOS33 } [get_ports {seg_common[1]}];
set_property -dict { PACKAGE_PIN G5    IOSTANDARD LVCMOS33 } [get_ports {seg_common[2]}];
set_property -dict { PACKAGE_PIN G4    IOSTANDARD LVCMOS33 } [get_ports {seg_common[3]}]; #MSB

set_property -dict { PACKAGE_PIN G2    IOSTANDARD LVCMOS33 } [get_ports {seg_data[7]}];#A
set_property -dict { PACKAGE_PIN G1    IOSTANDARD LVCMOS33 } [get_ports {seg_data[6]}];#B
set_property -dict { PACKAGE_PIN H5    IOSTANDARD LVCMOS33 } [get_ports {seg_data[5]}];#C
set_property -dict { PACKAGE_PIN H4    IOSTANDARD LVCMOS33 } [get_ports {seg_data[4]}];#D
set_property -dict { PACKAGE_PIN J5    IOSTANDARD LVCMOS33 } [get_ports {seg_data[3]}];#E
set_property -dict { PACKAGE_PIN J4    IOSTANDARD LVCMOS33 } [get_ports {seg_data[2]}];#F
set_property -dict { PACKAGE_PIN H2    IOSTANDARD LVCMOS33 } [get_ports {seg_data[1]}];#G
set_property -dict { PACKAGE_PIN H1    IOSTANDARD LVCMOS33 } [get_ports {seg_data[0]}];#DP
```

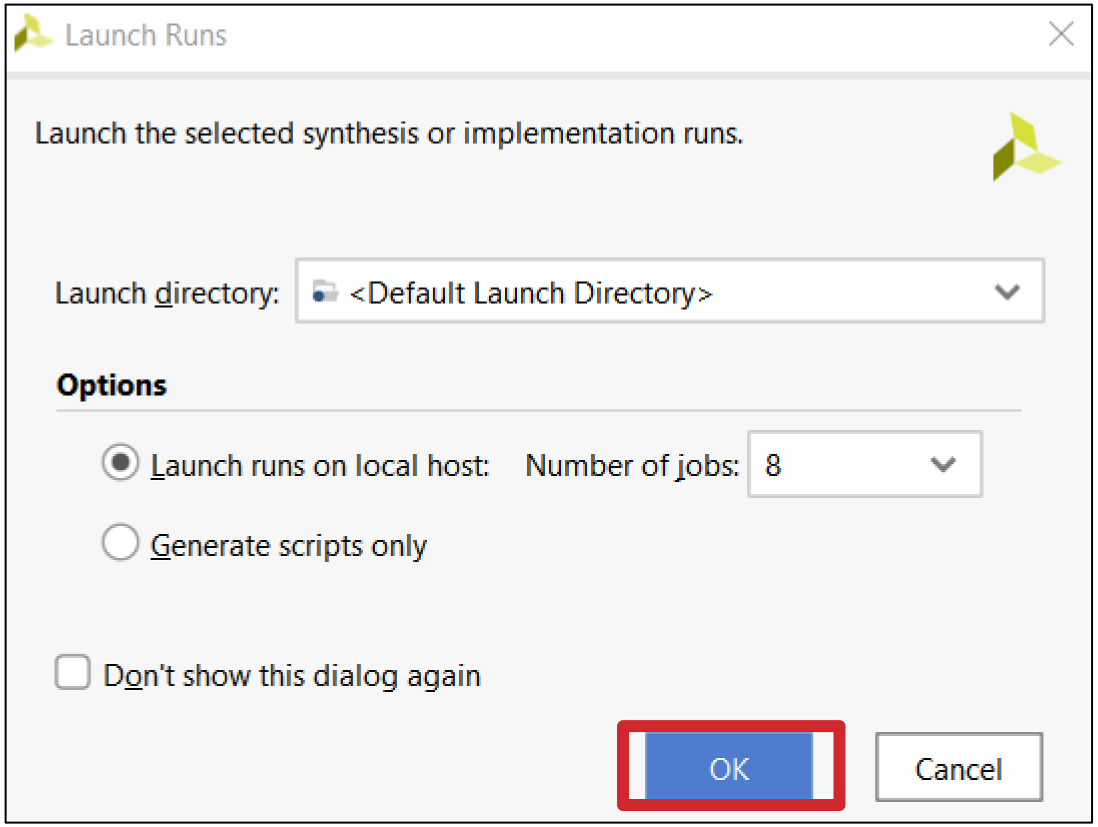
DESIGN EXAMPLE

- Vivado project
  - FPGA programming (mapping) – Implement Design



DESIGN EXAMPLE

- Vivado project
  - FPGA programming (mapping) – Implement Design


Launch Runs

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

Options

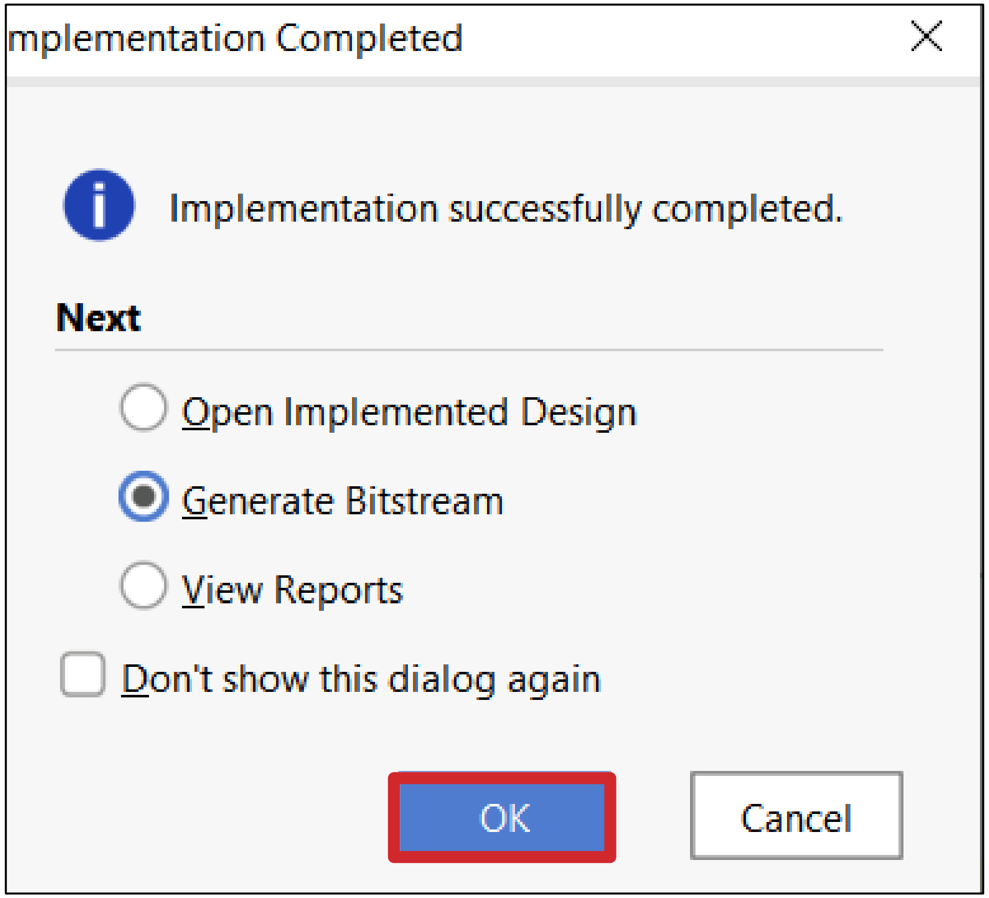
[✓ Launch runs on local host:] Number of jobs: 8

Generate scripts only

☐ Don't show this dialog again

DESIGN EXAMPLE

- Vivado project
  - FPGA programming (mapping) – Generate Programming File



mplementation Completed

Implementation successfully completed.

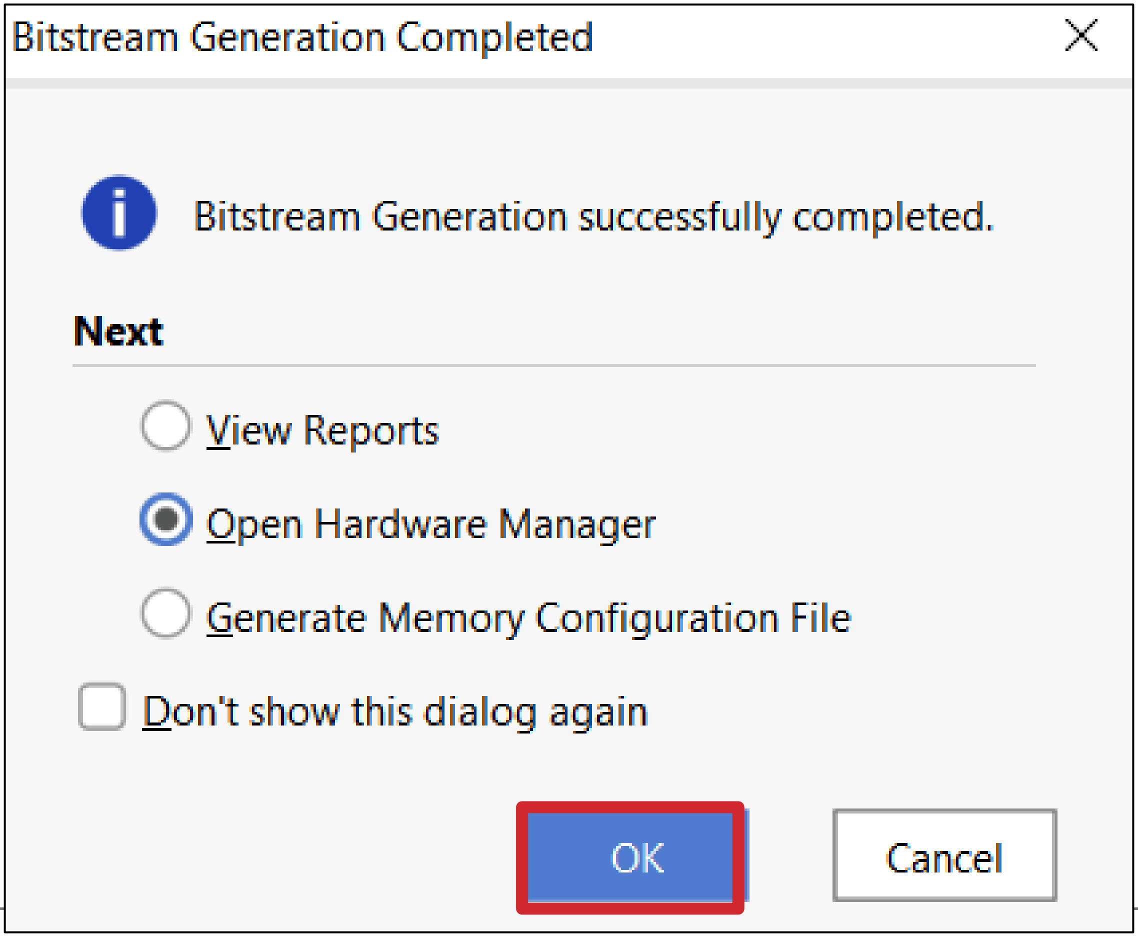
Next

- ○ Open Implemented Design
- ● Generate Bitstream
- ○ View Reports

☐ Don't show this dialog again

DESIGN EXAMPLE

- Vivado project
  - FPGA programming (mapping) – Generate Programming File



Bitstream Generation Completed

Bitstream Generation successfully completed.

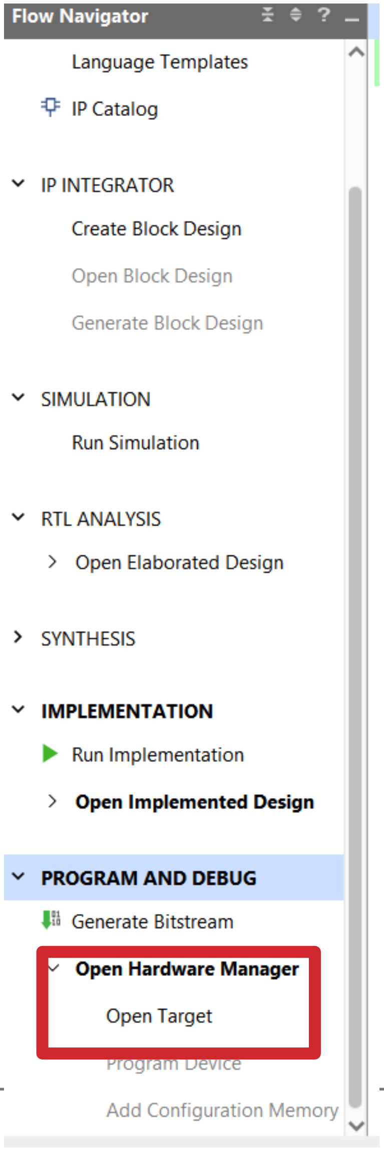
Next

- ○ View Reports
- ● Open Hardware Manager
- ○ Generate Memory Configuration File

☐ Don't show this dialog again

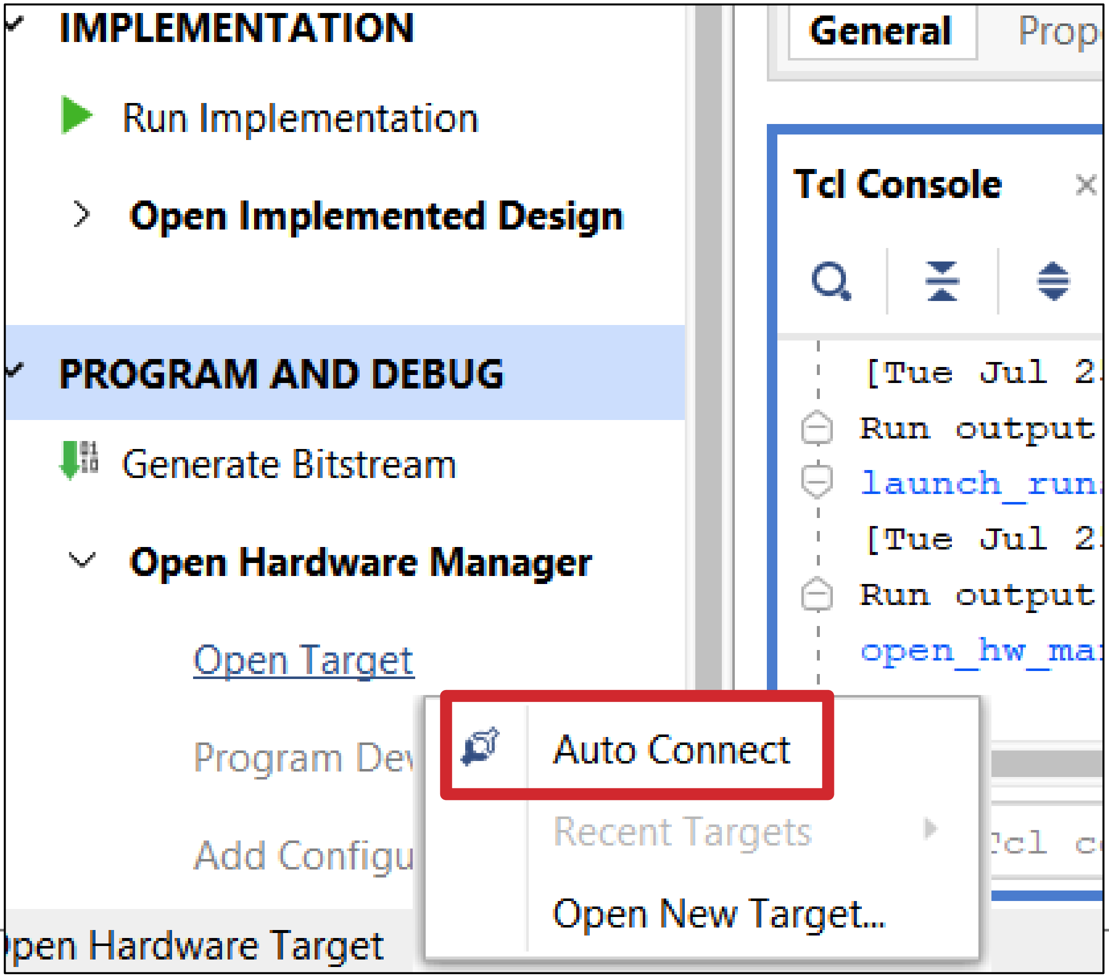
DESIGN EXAMPLE

- Vivado project
  - FPGA programming
    - Connect the FPGA board to the computer and click on Open Target



DESIGN EXAMPLE

- Vivado project
  - FPGA programming


IMPLEMENTATION

Run Implementation

Open Implemented Design

PROGRAM AND DEBUG

Generate Bitstream

Open Hardware Manager

Open Target

Program Dev

Add Configu

pen Hardware Target

Auto Connect

Recent Targets

Open New Target...

General

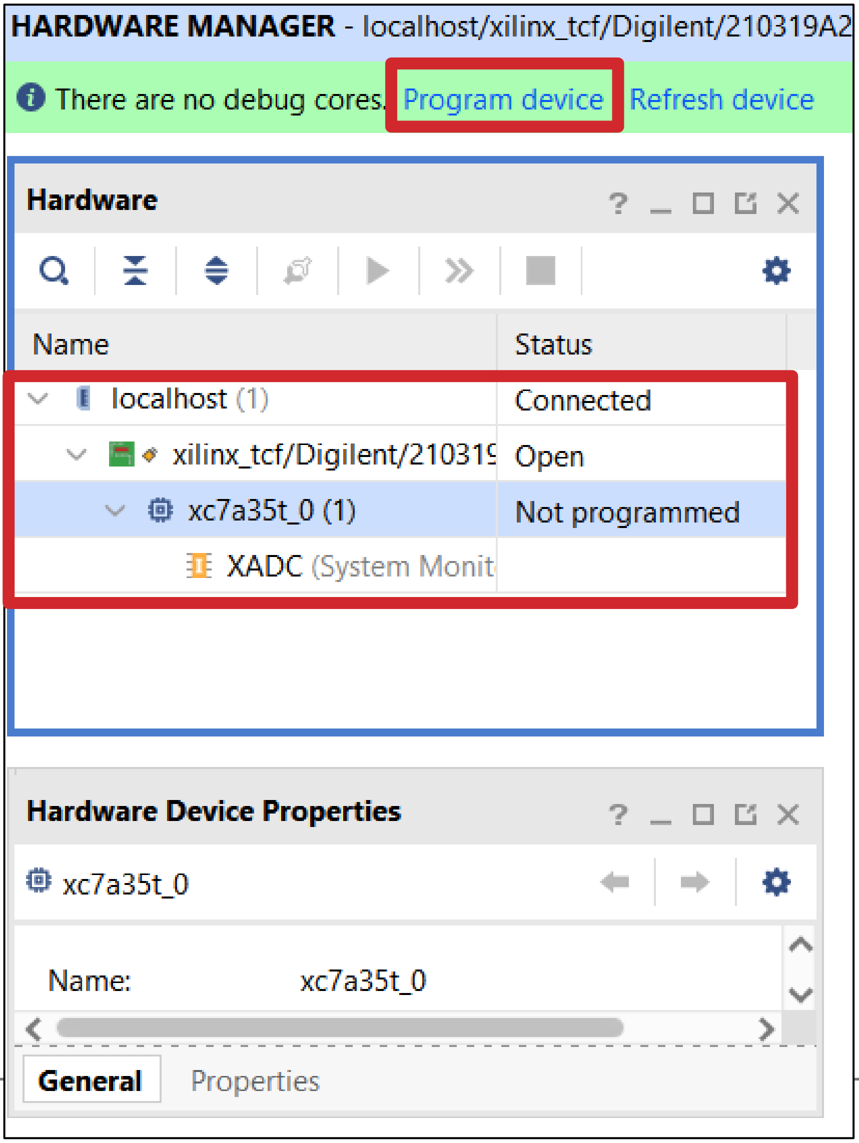
Prop

Tcl Console

```
[Tue Jul 2
Run output
launch_run
[Tue Jul 2
Run output
open_hw_ma
Tcl c
```

DESIGN EXAMPLE

- Vivado project
  - FPGA programming



HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210319A2

There are no debug cores. Program device Refresh device

Hardware

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210319	Open
xc7a35t_0 (1)	Not programmed
XADC (System Monit	

Hardware Device Properties

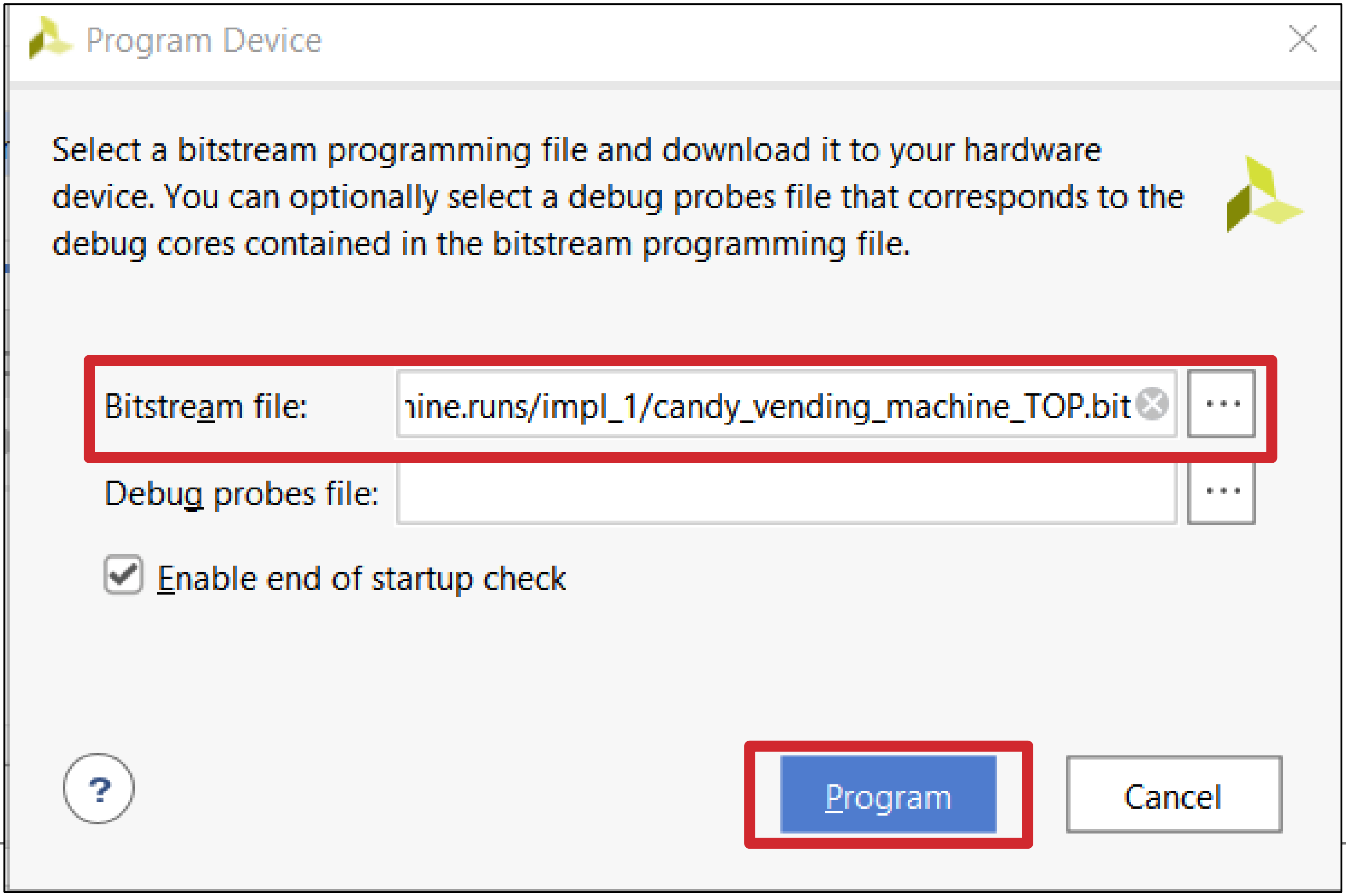
xc7a35t_0

Name: xc7a35t_0

General Properties

DESIGN EXAMPLE

- Vivado project
  - FPGA programming



Program Device

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: ıine.runs/impl_1/candy_vending_machine_TOP.bit

Debug probes file:

[✓] Enable end of startup check

DESIGN EXAMPLE

- Programming FPGA
  - Board Demo