Gillian Kearney

Lab Report 5

ECE 2031 L09

24 February 2022

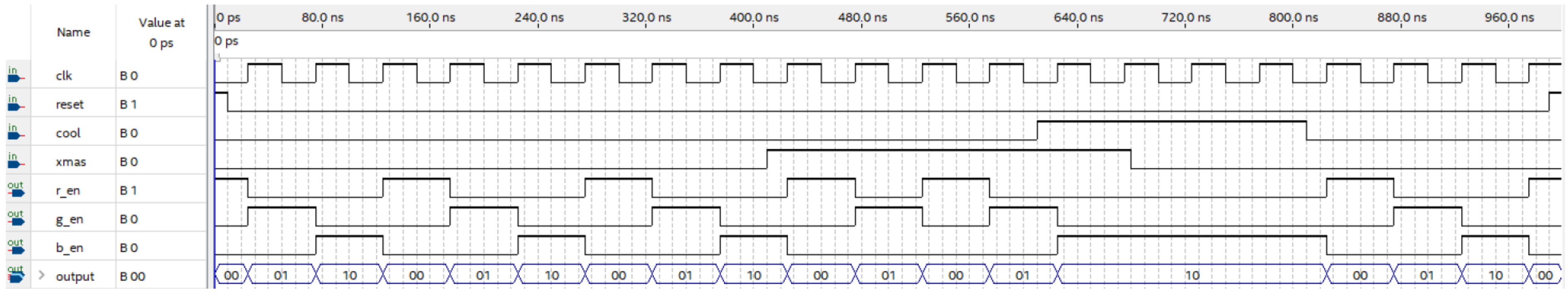
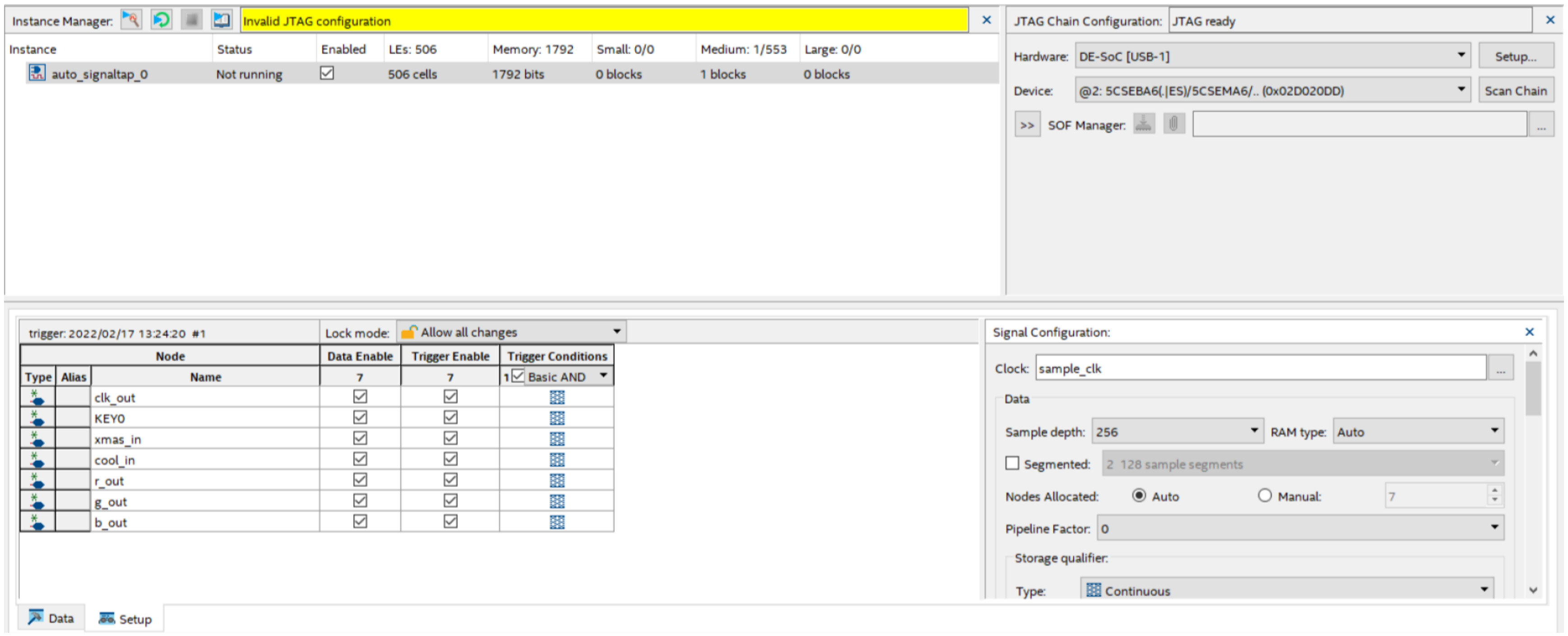


Figure 1. Functional simulation waveform of Red, Green, Blue finite state machine with 4 inputs and 4 outputs. The input vector covers all possible state transitions to prove the correctness of the circuit.



Instance Manager: Invalid JTAG configuration

Instance	Status	Enabled	LEs: 506	Memory: 1792	Small: 0/0	Medium: 1/553	Large: 0/0
auto_signaltap_0	Not running	✓	506 cells	1792 bits	0 blocks	1 blocks	0 blocks

JTAG Chain Configuration: JTAG ready

Hardware: DE-SoC [USB-1]

Device: @2: 5CSEBA6(.|ES)/5CSEMA6/.. (0x02D020DD)

SOF Manager:

trigger: 2022/02/17 13:24:20 #1

Lock mode: Allow all changes

Type	Alias	Name	Data Enable 7	Trigger Enable 7	Trigger Conditions 1 ✓ Basic AND
		clk_out	✓	✓	▨
		KEY0	✓	✓	▨
		xmas_in	✓	✓	▨
		cool_in	✓	✓	▨
		r_out	✓	✓	▨
		g_out	✓	✓	▨
		b_out	✓	✓	▨

Signal Configuration:

Clock: sample_clk

Data

Sample depth: 256 RAM type: Auto

Segmented: 2 128 sample segments

Nodes Allocated: [✓ Auto] Manual: 7

Pipeline Factor: 0

Storage qualifier:

Type: Continuous

Figure 2. Signal Tap configuration for specifications needed to read from the state machine implemented on the DE10-Standard Board.

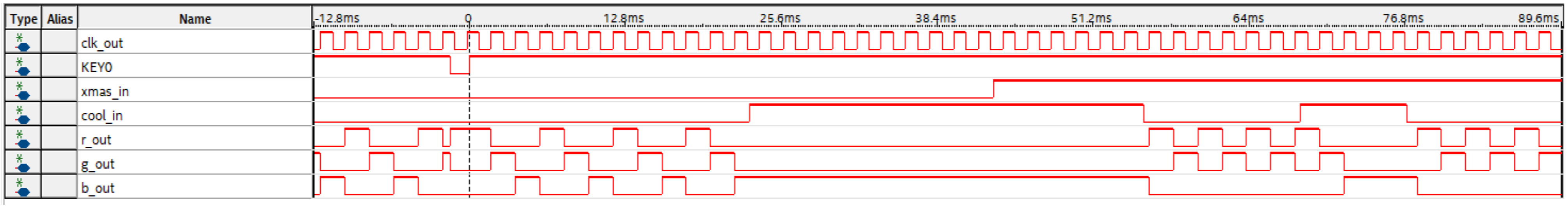


Figure 3. Signal Tap acquisition from a 3 output state machine implemented on the DE10 DE10-Standard Board. All possible state transition are achieved through variations of the input vector.

APPENDIX A

VHDL CODE IMPLEMENTING MOORE STATE MACHINE

```
-- StateMachine.vhd
-- Four-State Moore State Machine
-- Gillian Kearney
-- ECE 2031 L09
-- 2/24/22

-- A Moore machine's outputs are dependent only on the current state.
-- The output is written only when the state changes.  (State
-- transitions are synchronous.)

library ieee;
use ieee.std_logic_1164.all;

entity StateMachine is

	port(
		clk	: in  std_logic;
		reset	: in  std_logic;
		cool	: in std_logic;
		xmas	: in std_logic;
		output: out std_logic_vector(1 downto 0);
		r_en	: out std_logic;
		g_en	: out std_logic;
		b_en	: out std_logic
	);

end entity;

architecture rtl of StateMachine is

	-- Build an enumerated type for the state machine
	type state_type is (red, green, blue);

	-- Register to hold the current state
	signal state   : state_type;

begin

	-- Logic to advance to the next state
	process (clk, reset)
	begin
		if reset = '1' then
			state <= red;
		elsif (rising_edge(clk)) then
			case state is
				when red=>
					state <= green;
				when green=>
					if xmas = '1' and cool = '0' then
						state <= red;
					else
						state <= blue;
```

```
                              end if;
                         when blue=>
                              if cool = '0' then
                                   state <= red;
                              else
                                   state <= blue;
                              end if;
                    end case;
               end if;
          end process;

          -- Output depends solely on the current state
          process (state)
          begin
               case state is
                    when red =>
                         output <= "00";
                         r_en <= '1';
                         g_en <= '0';
                         b_en <= '0';
                    when green =>
                         output <= "01";
                         r_en <= '0';
                         g_en <= '1';
                         b_en <= '0';
                    when blue =>
                         output <= "10";
                         r_en <= '0';
                         g_en <= '0';
                         b_en <= '1';
               end case.
          end process;

     end rtl;
```