Digital Systems Design II

COEN 313 (Section FJ-X)

Experiment 5:

A Serial Two’s Complementer Usuing ASM Chart and

Dataphath/Control Unit Design Methodology

Andre Hei Wang Law

4017 5600

Performed on Monday, November 15, 2021

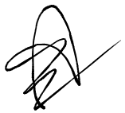
Due on Tuesday, December 14, 2021

“I certify that this submission is my original work and meets the Faculty’s Expectations of Originality”.

Tuesday, December 14, 2021

Andre Hei Wang Law

4017 5600



1) Objectives

For the final experiment of the course Coen 313, students have a main objective of becoming familiar with the ASM chart, control unit and datapath design methodology. Then, they will also learn to be well-versed in VHDL coding of datapath components and finite state machine at the RTL level.

2) VHDL Code

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_unsigned.all;

use IEEE.numeric\_std.all;

entity twos\_complementer is

port( din : in std\_logic\_vector(7 downto 0);

reset : in std\_logic ;

clk : in std\_logic ;

done\_out : out std\_logic ;

reg\_out : out std\_logic\_vector(7 downto 0));

end twos\_complementer ;

architecture arch\_twos of twos\_complementer is

type state\_type is (start, loading, lsb\_0, lsb\_1, invert);

signal state : state\_type;

signal d, shift, load, inc, clr, ld\_done, clr\_done, flip : std\_logic;

signal counter : std\_logic\_vector(2 downto 0);

signal shift\_register, temp\_reg : std\_logic\_vector(7 downto 0);

begin

process(reset, clk, counter, shift\_register)

begin

if (reset = '1') then

counter <= "000";

state <= start;

elsif (clk'event and clk='1') then

case state is

when start =>

counter <= "000";

shift\_register <= "00000000";

load <= '1';

shift <= '0';

inc <= '0';

clr <= '1';

ld\_done <= '0';

clr\_done <= '1';

flip <= '0';

state <= loading;

when loading =>

shift <= '0';

load <= '0';

clr <= '0';

clr\_done <= '0';

if (counter = "111") then

ld\_done <= '1';

done\_out <= '1';

else

if (flip = '1') then

inc <= '1';

state <= invert;

else

inc <= '1';

if (shift\_register(0) = '1') then

state <= lsb\_1;

else

state <= lsb\_0;

end if;

end if;

end if;

when lsb\_0 =>

d <= shift\_register(0);

inc <= '0';

shift <= '1';

flip <= '1';

state <= loading;

when lsb\_1 =>

d <= shift\_register(0);

inc <= '0';

shift <= '1';

flip <= '0';

state <= loading;

when invert =>

d <= not(shift\_register(0));

inc <= '0';

shift <= '1';

flip <= '0';

state <= loading;

when others =>

end case;

end if;

end process;

process (clk, inc, clr)

begin

if (clr = '1') then

counter <= "000";

clr <='0';

end if;

if (inc = '1') then

counter <= std\_logic\_vector(unsigned(counter) + 1);

end if;

end process;

process (clk, din, d, shift, load)

begin

if (load = '1') then

shift\_register <= din;

load <= '0';

end if;

if (shift ='1') then

temp\_reg <= shift\_register(6 downto 1) & d & shift\_register(7);

end if;

end process;

reg\_out <= not (temp\_reg);

process (clk, clr\_done, ld\_done)

begin

if (clr\_done = '1') then

done\_out <= '0';

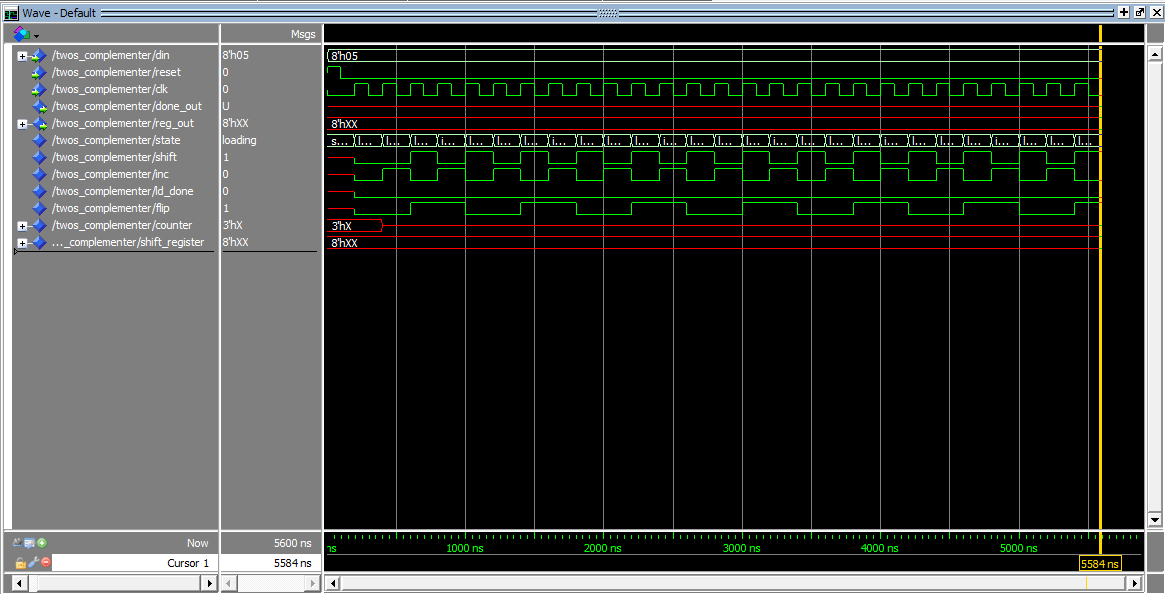
clr\_done <= '0';

end if;

end process;

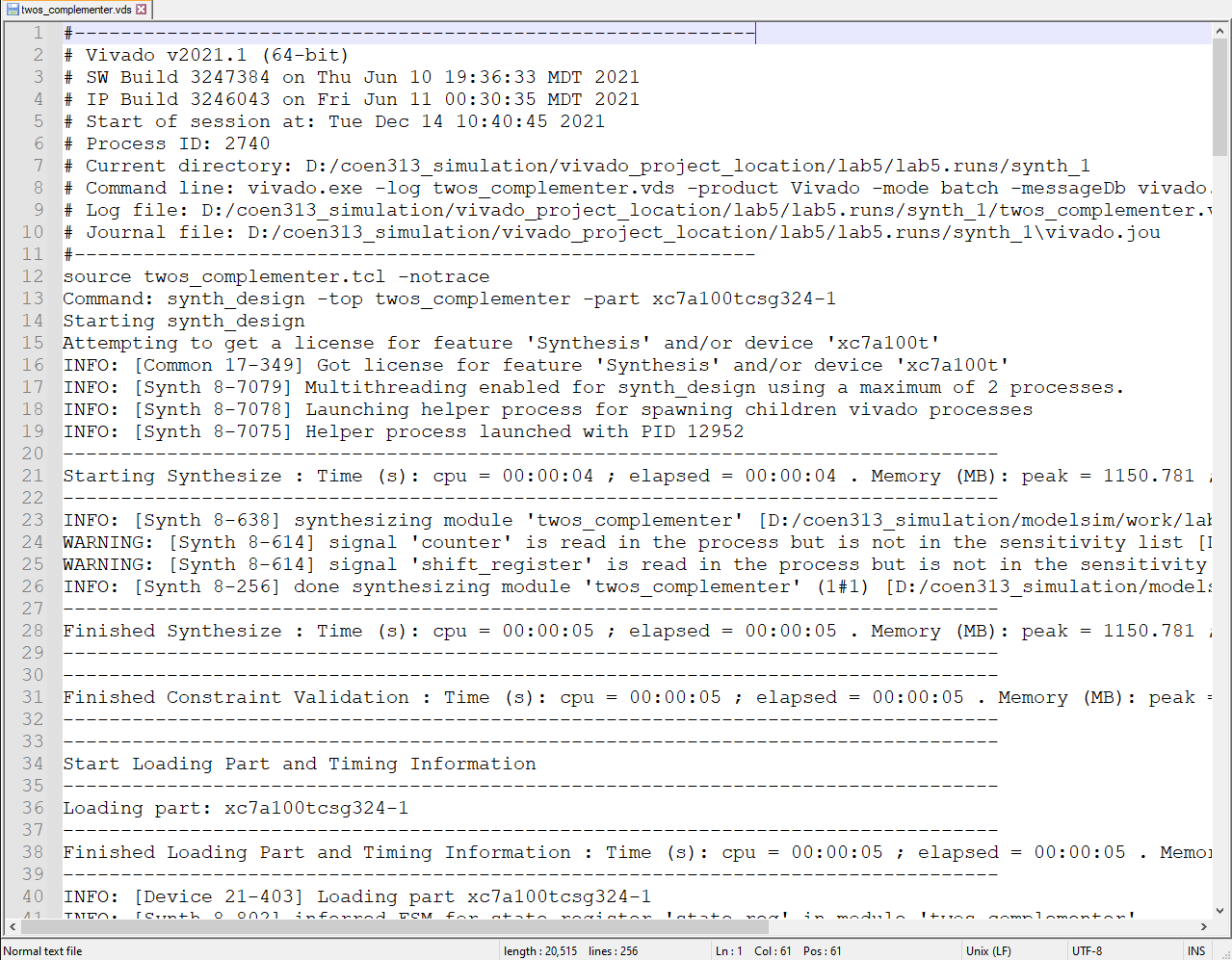
end architecture;

3)Simulation Results

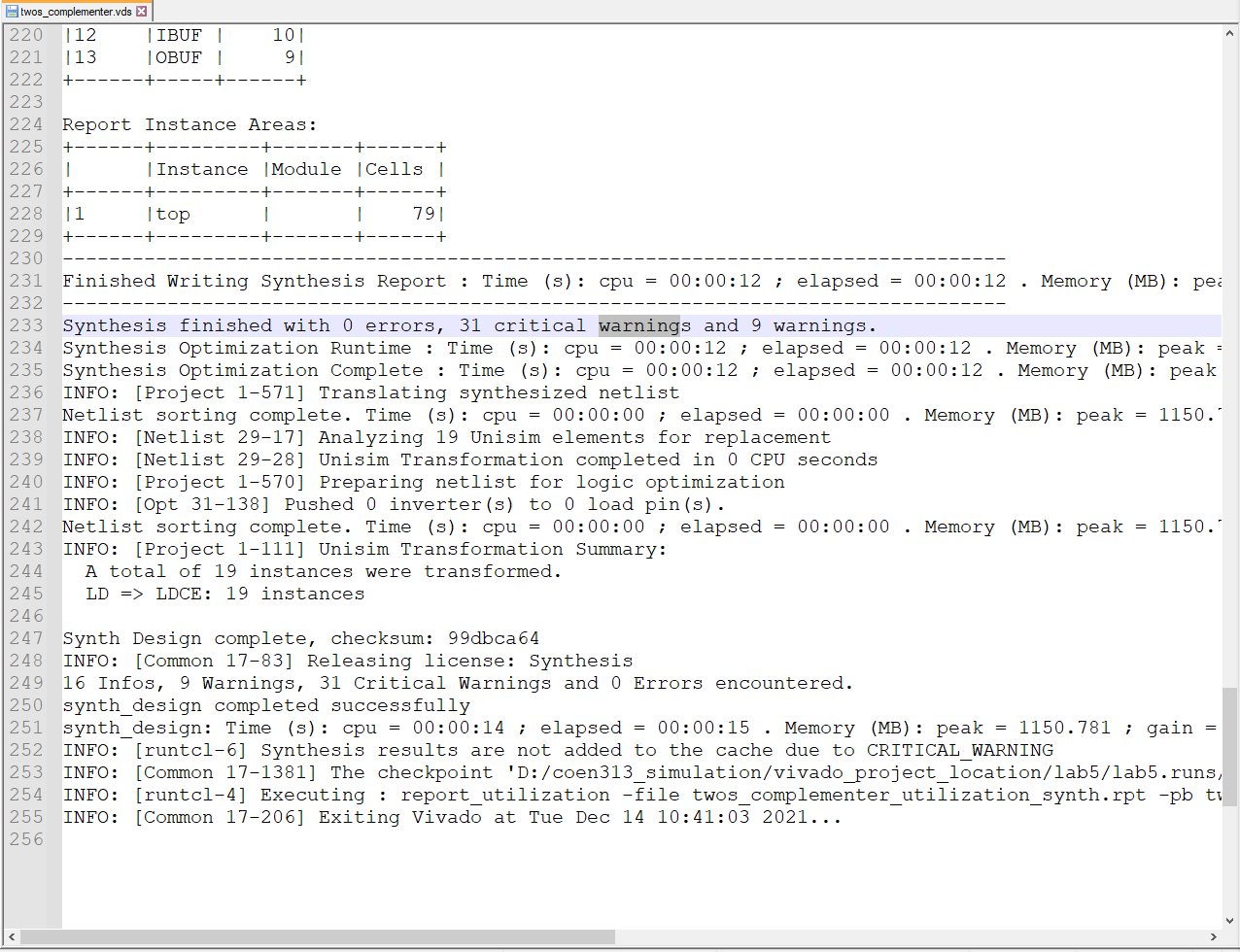


3.1 Modelsim Simulation Results

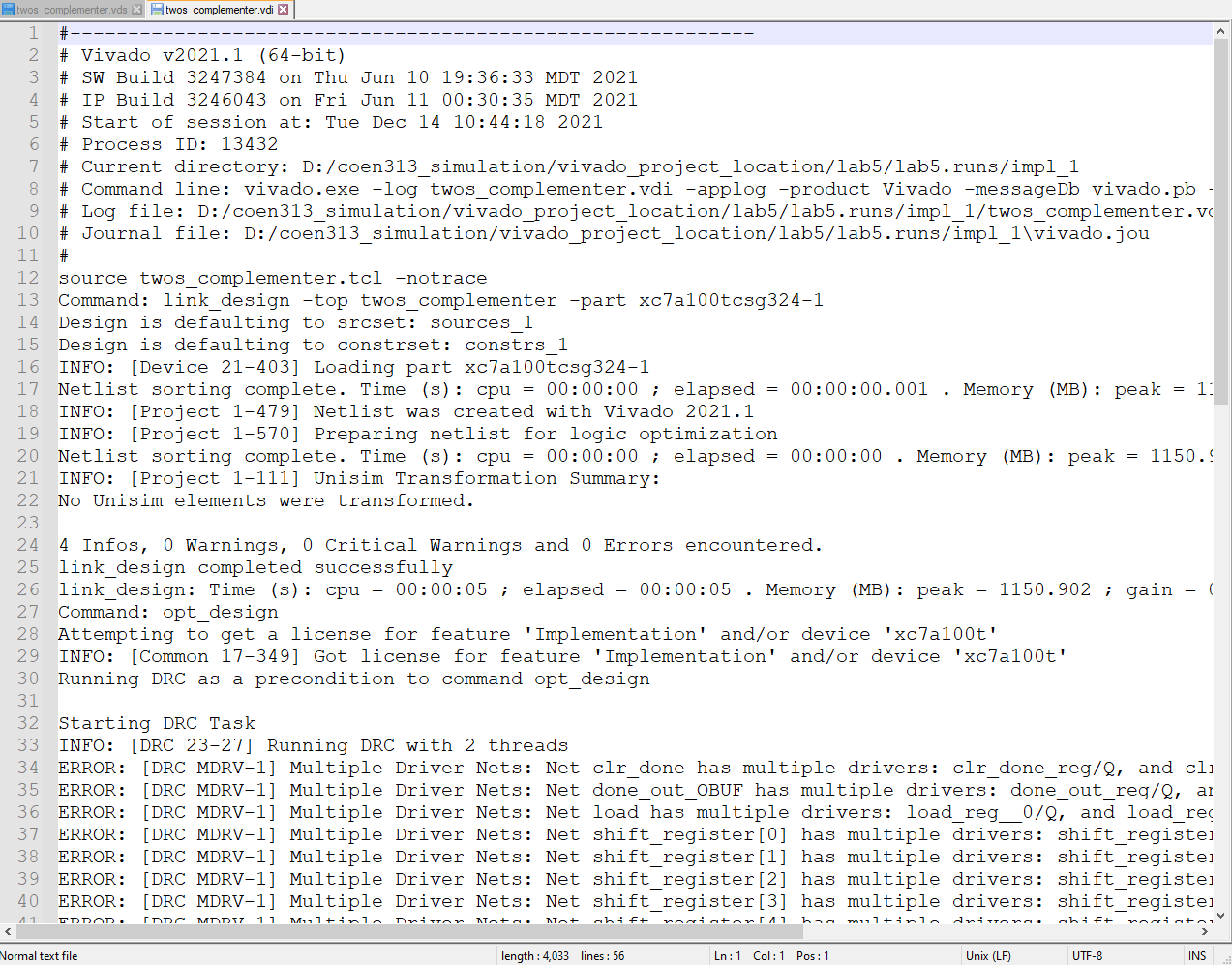
4) Synthesis and Implementation Log Files



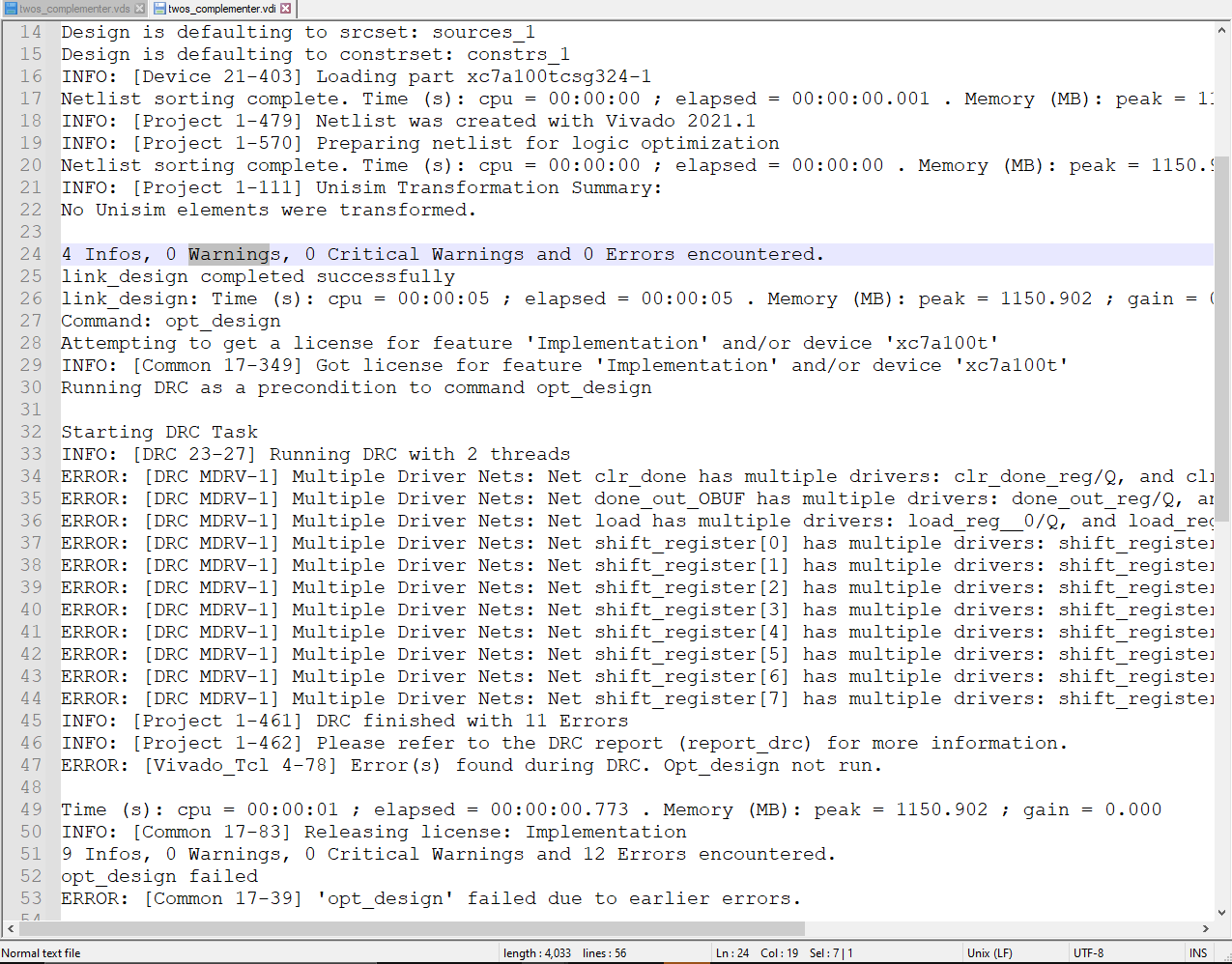
4.1.a Beginning of Synthesis Log File



4.1.b Example of a Synthesis Log File Warning Messages



4.2.a Beginning of Implementation Log File



4.2.b Example of an Implementation Log File Warning Messages

The messages, errors and warnings generated by Vivado can be useful in pinpointing problems and help explaining why this area has an issue. These log files are found in the Vivado project directory in which the synthesis log file is under “/.runs/synth\_1/entity\_name.vds” while the implementation log file is under “/.runs/impl\_1/entity\_name.vdi”. The above figure 4.1.b and 4.2.b are some examples of warning messages encountered during implementation and synthesis, respectively. The importance of these generated messages or warning is to aide the user to locate potential problems and solve them easily by reading the type of error messages. In the long run, unresolved minor warnings can cause unwanted results, thus promoting the user to always solve their “warnings”, “critical warning” and “errors encountered”. The best-case scenario is when the log file indicates that there are zero problems. Notice that in this case, the Modelsim simulation results had issues while still being able to be synthesised and implemented. The cause of the unwanted red line (no signal) from figure 3.1 can be explained by the many critical warnings and warnings such as shown in figure 4.1.b. Overall, it is a good practice to revise the VHDL code in order to solve all these errors, critical warnings and warnings.

5) Bit File



6.1 Bit File of the Two’s Converter (from NotePad++)

6) Questions

6.1) Examine the .vds log file contained in the synth\_1 directory created by the Xilinx Vivado tools. Your design contains a finite state machine. Comment on the state encoding used by the synthesis tool.

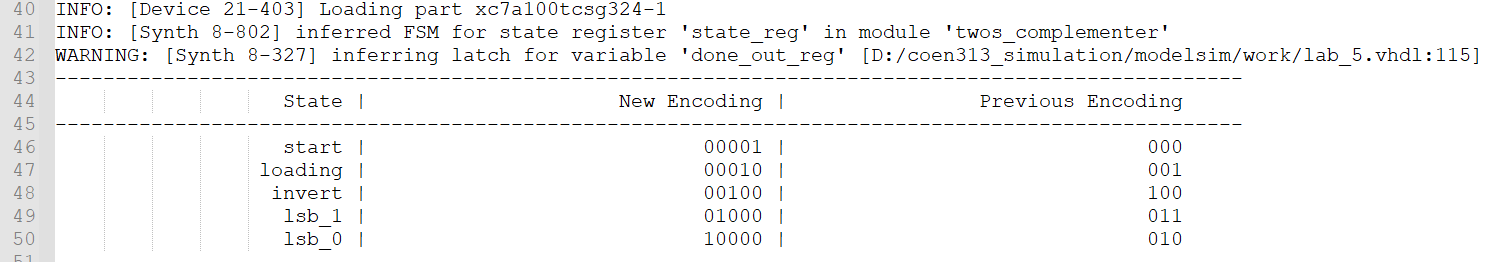


Figure 6.1 Encoding State from the Synthesis Log File

This is a one-hot state encoding. This can be deduced based on the fact that the “new encoding” section has as many encodings as there are of bits in which only one of them are “ON” while the rest remains “OFF”.

6.2) Comment on the advantages/disadvantages of a binary state encoding vs. a one-hot state encoding. Do other state encoding techniques exist?

The advantage of binary state encoding is that the encoding is a straightforward. The assigned values go sequentially to the states in which as few bits are used to encode these states. On the other hand, a one-hot state encoding uses one ON bit and remaining OFF bits to represent the current state. The overall total of bits used is excessively larger than the binary state encoding. However, its strength lies with its simplification where the stimulus logic for the flip flips doesn’t need to be decoded as the bits itself are the states. Yes, there exists another encoding technique and it is called the “Gray Encoding” which consists of a sequency of one-bit changes between one value to the next one. Its advantage is that this technique uses less amounts of bits.

6.3) Consider a datapath which contains a register which has a control signal called “LOAD\_MICK” which acts as a synchronous load control signal for the register. Explain by means of a timing diagram and in words the error contained in the following ASM chart (Figure 3):

The error of figure 3 is that “LOAD\_MICK” should not be inside Keith while its condition expression, “MICK = 6”, is below. Rather, we should load the signal first by means the synchronous load control signal, then go through the Keith state and its condition expressions.