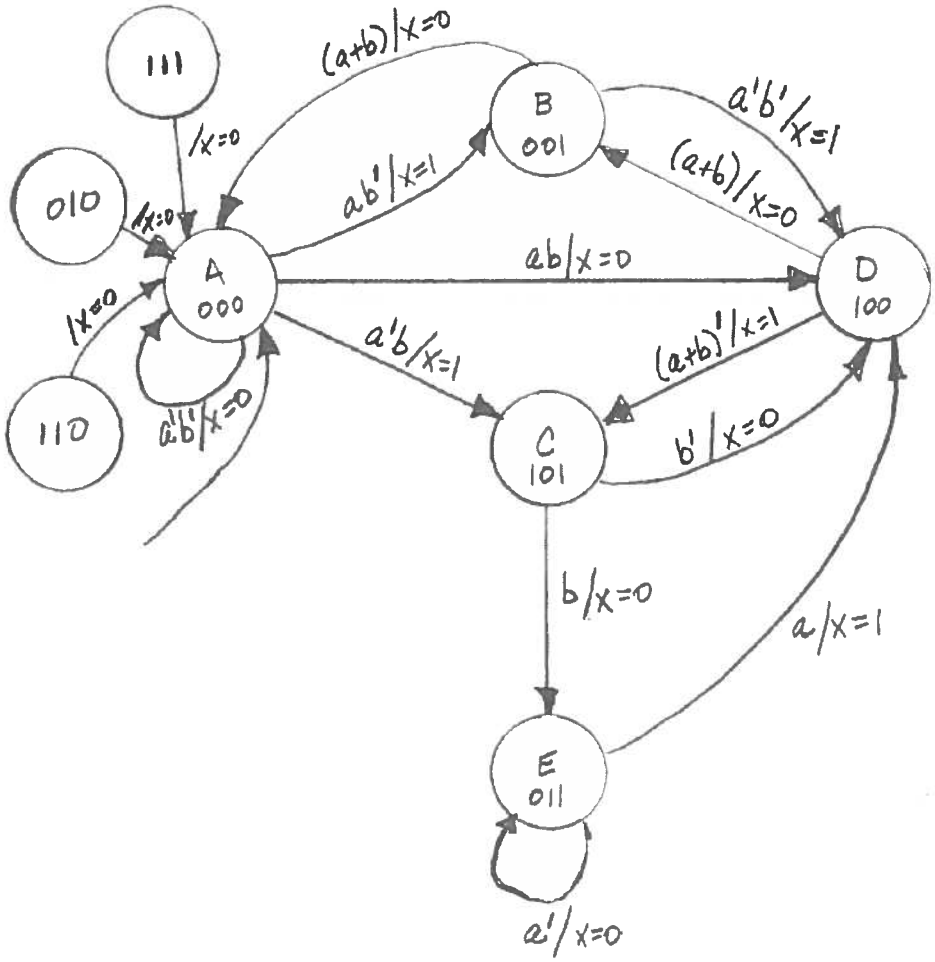
NAME SOLUTION **UIN** _____

Final Exam

ECEN 449/749
December 13, 2022

1. (33% Undergrad / 25% Grad) The Verilog code representing the Finite State Machine (FSM) shown below is given on the following pages. There is missing Verilog code in four places that are boxed and labeled (a) through (d). Write the Verilog statement(s) to go where the question marks (i.e., ??) are in the code. The FSM below is a Mealy machine with Mealy output **x**.

Inputs: a, b (bit)
Output: x (bit)



(a) (8% Undergrad / 6% Grad)

```
input b
reg x
  c = 3'b101
  reg [2:0]
```

(b) (8% Undergrad / 6% Grad)

```
x = 1
Next State = B
```

(c) (8% Undergrad / 6% Grad)

```
if (w(a|b) == 1)
    x = 1
    Next State = C
```

(d) (9% Undergrad / 7% Grad)

```
posedge clk
if (est == 1) State <= A
else  State <= Next State
```

```
module FinalExam_Fall_2022 (x,a,b,Clk,Rst);
  output x;
  input a, ??, Clk, Rst;
  reg ??;

  parameter A=3'b000, B=3'b001, C=3'b???, D=3'b100,
            E=3'b011;
  reg [??:0] State, NextState;
```

(a)

```
//Combinational Logic
   always @(State, a, b)
   begin
     case(State)
       A: begin
            if( (~a & ~b) == 1) begin
                x = 0;
                NextState = A;
            end
            else if( (~a & b) == 1) begin
                x = 1;
                NextState = C;
            end
            else if( (a & b) == 1) begin
                x = 0;
                NextState = D;
            end
            else begin
                x = ??;
                NextState = ??;
            end
          end
```

(b)

```
B: begin
       if( (a | b) == 1) begin
           x = 0;
           NextState = A;
       end
       else if( (~a & ~b) == 1) begin
           x = 1;
           NextState = D;
       end
    end
C: begin
       if( ~b == 1) begin
           x = 0;
           NextState = D;
       end
       if( b == 1) begin
           x = 0;
           NextState = E;
       end
    end
D: begin
       if( (a | b) == 1) begin
           x = 0;
           NextState = B;
        end
       if( ~(a | b) == ??) begin
           x = ??;
           NextState = ??;
       end
    end
```

(c)

```
        E: begin
              if( ~a == 1) begin
                   x = 0;
                   NextState = E;
              end
              else if( a == 1) begin
                   x = 1;
                   NextState = D;
              end
            end
   Default: begin
              x = 0;
              NextState = A;
              end
      endcase
     end // end of combinational always @
//Sequential
   always @(?? Clk)
   begin
     if(Rst == 1) State ?? ??;
     else State ?? ??;
   end // end of sequential always @
endmodule
```

(d)

2. (33% Undergrad / 25% Grad) In this problem we will consider the Free Lossless Audio Codec (FLAC) for encoding and storing five channel audio. FLAC playback is supported by Windows 10/11, MacOS, Linux, Android 3.1 and later, and iOS 11 and later. There are a number of applications that will transcode various audio formats into the FLAC format.

For parts (a) and (b) below we will be considering FLAC with five audio channels (Left-front, Right-front, Center, Surround-left, and Surround-right), 16-bit resolution, and 48 kHz sampling rate, which is referred to as "Five-channel, 16-bit/48 kHz."

(a) (8% Undergrad / 6% Grad) In the FLAC five channel, 16-bit/48 kHz format, each of the five channels is sampled with a 16-bit Analog-to-Digital (A/D) converter, and the result is stored as a 16-bit, two's complement binary number. What is the combined information transfer rate for all five channels in bits per second without any compression or FLAC message overhead (i.e., the total bit rate to send the left-front, right-front, center, surround-left, and surround-right channels)?

Total bit rate = 5 chans × 16 bits/sample × $48 \cdot 10^3$ samples/sec

= 3,840,000 bps

= 3.84 Mbps

(b) (25% Undergrad / 19% Grad) The Free Lossless Audio Codec (FLAC) format is similar to MP3, but it is *lossless*, which means that audio compressed with FLAC has no loss in quality. This is similar in concept to the lossless compression schemes we mentioned in class (e.g., Huffman Coding, Run Length Encoding, Lempel-Ziv Coding), but we get better compression because FLAC is designed for audio. FLAC typically compresses audio to between 40% to 60% the size of the

uncompressed audio. In this problem, we will look at the FLAC format but not the details of the compression algorithm, which uses a Linear Predictive Coding (LPC) model to approximate the speech signal in such a way that when the approximation is subtracted from the actual audio signal, the result (called the residual or error) requires fewer bits-per-sample to encode.

The FLAC format for a stream (where a stream is a track/song) consists of the following: (1) **Four bytes** consisting of the ASCII characters "fLaC" that mark the beginning of the stream; (2) a STREAMINFO block that consists of **34 bytes** with information about the entire stream such as the sample rate in Hz, number of channels, bits per sample, and total number of samples in the stream; and (3) one or more Audio Frames (described in the next paragraph).

Audio Frames in a FLAC stream with five channels (Left-front, Right-front, Center, Surround-left, and Surround-right) and a block size of 2048 audio samples have the following elements: (1) Frame Header of **five bytes** that starts with a sync code and contains the minimum information for a decoder to play the stream, including the block size in samples, a repeat of the sample rate and bits per sample, and CRC-8 error detection; (2) Sub-Frame Header Left-front of **one byte**; (3) Audio Samples for the left-front channel of size **2048 samples** (each sample is 16 bits with **no** compression and less than 16 bits on the average with compression); (4) Sub-Frame Header Right-front Channel of **one byte**; (5) Audio samples for the right-front channel of size **2048 samples** (each sample is 16 bits with **no** compression and less than 16 bits on the average with compression), (6) Sub-Frame Header Center Channel of **one byte**; (7) Audio samples for the center channel of size **2048 samples** (each sample is 16 bits with **no** compression and less than 16 bits on the average with compression); (8) Sub-Frame Header Surround-left of **one byte**; (9) Audio Samples for the surround-left channel of size **2048 samples** (each sample is 16 bits with **no** compression and less than 16 bits on the average with compression); (10) Sub-Frame Header Surround-right of **one byte**; (11) Audio Samples for the surround-right channel of size **2048 samples** (each sample is 16 bits with **no** compression and less than 16 bits on the average with compression); (12) CRC-16 (**two bytes**) for error dection on the Audio Frame.

(i) (6% Undergrad / 4% Grad) How many Audio Frames do you need for a 1 second audio stream? You will want to round up to the next integer number of audio frames if you have a fractional part (e.g., if you calculated 253.1 audio frames, your answer should be 254). Show your work for partial credit.

$$\text{Audio Frames/sec} = \left\lceil \frac{1\cdot\text{sec} \cdot 48\cdot 10^3 \text{ samples/sec}}{2048 \text{ samples/audio frame}} \right\rceil$$

$$= \left\lceil \frac{48\cdot 10^3}{2048} \right\rceil = \lceil 23.44 \rceil = 24/\text{sec}$$

(ii) (6% Undergrad / 5% Grad) How many bits make up a 1 second FLAC audio stream with **no** compression? Assume that the last frame has the same number of audio samples as all the other frames (i.e., the last frame is padded with samples with a value of 0 to fill up the last frame). Show your work for partial credit.

FLAC Stream = | fLaC | Streaminfo | Audio Fram#1 | ... | Audio Frame 24 |

Audio Frame #i = | 5B Frame Hdr | 1B L-f | 2048 x 2B L-f samp | 1B R-f | 2048 x 2B L-f samp |

| 1B Center | 2048 x 2B Center | 1B S-l | 2048 x 2B S-l samp |

| 1B S-R | 2048 x 2B S-R | 2B CRC-16 |

$= 5\text{B} + 5\left(1\text{B} + 4096\text{B}\right) + 2\text{B} = 5 + 5 + 5\times 4096 + 2$

$= 12 + 20{,}480 = 20{,}492 \text{ B/Audio Frame}$

FLAC Stream = $4\text{B} + 34\text{B} + 24 \times 20{,}492 = 491{,}846$ B/sec

bps = $491{,}846 \times 8$ bits/B $= 3{,}934{,}768$ bits/1 sec

(iii) (6% Undergrad / 5% Grad) How many *overhead bits* are transmitted in the 1 second uncompressed FLAC stream in part (ii) above? *Overhead bits* are any bits transmitted in the stream that are *not* audio samples. Do not count the padded samples of 0 in the last frame as overhead bits. Show your work for partial credit.

FLAC Stream Overhead = 4B + 34B + 24 Audio frames/sec × (5B + 5×1B + 2B)
flaC streaminfo
= (4 + 34 + 24×12)B = 326 B/sec × 8 bits/B
= **2,608 bits of overhead in one sec**

Aside: very small overhead $\frac{2608}{3,934,768} = 6.63 \cdot 10^{-4} = 0.0663\%$

(iv) (7% Undergrad / 5% Grad) Assume after lossless FLAC compression that the number of bits required to store the 2048 samples for each of the five channels in an Audio Frame is only 40% the number of bits with **no** compression. You will want to round up to the next integer number of bytes per audio channel if you have a fractional part (e.g., if you calculated 253.1 bytes per audio channel, your answer should be 254). What is the average bitrate in bits per second for the compressed FALC stream? (*Hint:* You cannot just take 40% of your answer in part (ii).).

Each Audio Frame = |5B Hdr| 1BL | ⌈0.4 × 2048 × 2 B⌉ | ... | 2B CRC |
= 5B + 5 ↑ (5 chans) (1 + ⌈1,638.4 B⌉) + 2
= 5B + 5(1 + 1639)B + 2B = 8207

FLAC 1sec Compressed stream = 4B + 34B + 24 × 8207
= 197,006 B

197,006 B × 8 bits/B = **1,576,048 bps**

3. (34% Undergrad / 25% Grad) Consider a digital design that needs to process data that arrives at a rate of 10 MHz.

   You have implemented a prototype of the design in software on a microprocessor, and it takes $260 \cdot 10^{-9}$ sec for the prototype to process the data. You find that the software prototype has two functions: F1 and F2 that take up 35% and 30% of the prototype runtime, respectively.

   If you implement both the functions F1 and F2 on a **single** FPGA, you will obtain a speedup of 10X and 20X, respectively. The speedup includes the interconnect delay in communications between the FPGA and the microprocessor.

   If you implement the functions F1 and F2 on their own FPGA's (i.e., F1 is on one FPGA and F2 is on a second FPGA), you will obtain a speedup of 20X and 40X, respectively. By dedicating more FPGA resources to each fuction, we are able to improve performance by additional paralleism. The speedup includes the interconnect delay in communications between the FPGA(s) and the microprocessor.

   Answer parts (a) through (e) below on various combinations of software and FPGA(s).

   (a) (4% Undergrad / 3%Grad) Can the design be implemented purely in software and meet the design requirement? Show your work for partial credit.

   $$\frac{1}{260 \cdot 10^{-9}\,\text{sec}} = 3{,}846\ \text{MHz} < 10\ \text{MHz}$$

   $\therefore$ No, cannot meet the design requirement

   (b) (7% Undergrad / 5% Grad) Suppose you implement both F1 and F2 on a **single** FPGA. What will be the max data rate that can be sustained in MHz? Does this meet the design requirement? Show your work for partial credit.

   $$S(s_1, s_2) = \frac{1}{(1-0.35-0.3) + \frac{0.35}{10} + \frac{0.3}{20}} = \frac{1}{0.35+0.035+0.015}$$

   $$= \frac{1}{0.4} = 2.5\times$$

   $$\frac{260 \cdot 10^{-9}}{2.5} = 104 \cdot 10^{-9} \quad \frac{1}{104 \cdot 10^{-9}} = 9.62\ \text{MHz} < 10\ \text{MHz}$$

   $\therefore \Rightarrow$ No

(c) (7% Undergrad / 5% Grad) Suppose you implement F1 in software on the microprocessor and F2 in a dedicated FPGA. What will be the max data rate that can be sustained in MHz? Does this meet the design requirement? Show your work for partial credit.

$$S(s1) = \frac{1}{(1-0.3) + \frac{0.3}{40}} = \frac{1}{0.7 + 0.0075} = 1.4134$$

$$\frac{260 \cdot 10^{-9}}{1.4134} = 183.95 \cdot 10^{-9}\,\text{sec}, \quad \frac{1}{183.95 \cdot 10^{-9}\,\text{sec}} = 5.436\,\text{MHz} < 10\,\text{MHz}$$

∴ No

(d) (7% Undergrad / 5% Grad) Suppose you implement F2 in software on the microprocessor and F1 in a dedicated FPGA. What will be the max data rate that can be sustained in MHz? Does this meet the design requirement? Show your work for partial credit.

$$S(s1) = \frac{1}{(1-0.35) + \frac{0.35}{20}} = \frac{1}{0.65 + 0.0175} = 1.4981$$

$$\frac{260 \cdot 10^{-9}}{1.4981} = 173.55 \cdot 10^{-9}\,\text{sec}, \quad \frac{1}{173.55 \cdot 10^{-9}\,\text{sec}} = 5.76\,\text{MHz} < 10\,\text{MHz}$$

∴ No

(e) (9% Undergrad / 7% Grad) Suppose you implement F1 on a dedicated FPGA and F2 on a dedicated FPGA. What will be the max data rate that can be sustained in MHz? Does this meet the design requirement? Show your work for partial credit.

$$S(s1, s2) = \frac{1}{(1-0.35-0.3) + \frac{0.35}{20} + \frac{0.3}{40}} = \frac{1}{0.35 + 0.0175 + 0.0075}$$

$$= \frac{1}{0.375} = 2.6667$$

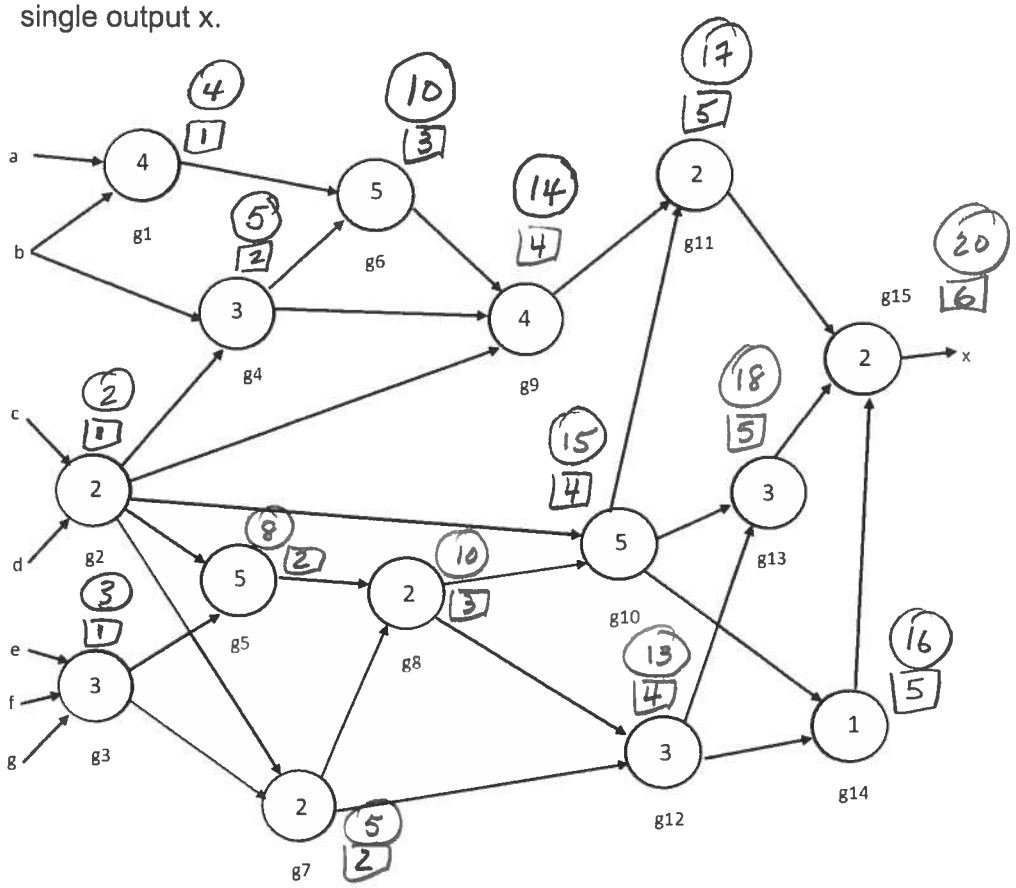
$$\frac{260 \cdot 10^{-9}}{2.6667} = 97.5 \cdot 10^{-9}\,\text{sec}, \quad \frac{1}{97.5 \cdot 10^{-9}\,\text{sec}} = 10.26\,\text{MHz} > 10\,\text{MHz}$$

Yes

4. (25% Grad) **GRAD Problem.** In this problem you will perform Static Timing Analysis (STA) of the circuit below. Each circle represents a logic gate, and each gate is given a name (i.e., g1, g2, …, g15). The number inside the circle is the delay of the gate in nano seconds. The seven inputs (i.e., a, b, c, d, e, f, and g) can all change at time 0. There is a single output x.

□ = level

○ = delay



(a) (10% Grad) Perform a levelization of the circuit above and fill in the following table with a list of the gates at each level. All of the gates should appear in the table.

Level	Gates
Level 1	g1, g2, g3
Level 2	g4, g5, g7
Level 3	g6, g8
Level 4	g9, g10, g12
Level 5	g11, g13, g14
Level 6	g15
Level 7	—

(b) (10% Grad) Find the maximum delay at the output of each gate and fill in the table below.

Gate	Maximum Delay in nano seconds
g1	4
g2	2
g3	3
g4	5
g5	8
g6	10
g7	5
g8	10
g9	14
g10	15
g11	17
g12	13
g13	18
g14	16
g15	20

(c) (5% Grad) What is the maximum delay for the circuit (i.e., the max delay to x) based on your work in part (b)?

20 nsec