EECS 373 *Final*

Winter 2016

Name: _____ unique name: _____

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so. Nor did I discuss this exam with anyone after it was given to the rest of the class.

_____

NOTES:

1. Closed book/notes.
2. There are 15 pages including this one. ***The last page is a reference sheet. You may wish to rip it out.***
3. Calculators are allowed, but no PDAs, Portables, Cell phones, etc. Nothing capable of wireless communication!
4. Don't spend too much time on any one problem. If you get stuck, move on!!!
5. You have about 120 minutes for the exam.
6. **Be sure to show work and explain what you've done when asked to do so.**
   - Getting partial credit without showing work will be rare.
7. **If you use the back of a page or write your answer on a page other than where the answer is, be sure that is *clearly* noted on the page where the answer belongs.**

1. Multiple choice/fill-in-the-blank **[8 points, -2 per wrong or blank answer, minimum 0]**

    a. Of Zigbee, Bluetooth, and Wi-fi, _____ has the longest range.

    b. Of Zigbee, Bluetooth, and Wi-fi, _____ has the highest data rate.

    c. Say you find that the Wii remote doesn't work particularly well outside. Why might that be?
        - The magnetometer in the Wii remote would need metal objects near by
        - The IR camera in the Wii remote wouldn't work well in sunlight
        - The ultrasonic distance sensor in the Wii remote would struggle with wind
        - The gyroscope in the Wii remote won't work without large, flat objects (like walls) nearby.

    d. What is the primary advantage of a stepper motor over a DC brushed motor?
        - You have a lot more control over absolute position of a stepper motor
        - Stepper motors are generally cheaper
        - The max speed of a stepper motor is generally higher than that of a DC motor.
        - Steppers are easier to control than a DC motor.

    e. "Indoor localization" is trickier than "outdoor localization" because:
        - IR distance sensors work better outdoors than indoors.
        - Wi-fi travels much farther outside.
        - GPS doesn't work well (if at all) indoors.
        - It's easier to harvest energy outside than inside (mostly due to sun vs. natural lights).

2. Higher-level questions **[10 points]**
    a. USB was developed to replace the old parallel and serial ports on PCs. Among other things, it provided a higher data rate than serial (UART) or parallel ports. But unfortunately, USB is quite tricky to work with (complex protocol and non-standard signaling). In embedded systems, the typical way to deal with this complexity is to use a USB-to-UART converter. But clearly *that* just gets us back to UART speeds. ***Give two reasons*** why we don't just user UART rather than going to USB and then converting to UART. **[4]**

    b. Explain why energy harvesting is important to the future of the "Internet of Things*".* *Your answer must be 40 words or less.* **[4]**

    c. Explain why embedded security can be more *important* than "traditional" computer security (on a PC or server). **[3]**

3. You've been put in charge of designing an altitude-measuring device. You have an air-pressure sensor that can be used to compute altitude. It outputs 3 volts at sea-level and 1 volt if 4,000m above sea level. Its response is ***linear*** within that range (so 2 volts at 2,000m above sea level) and you can assume it has infinite resolution. Your company wants your device to output the current altitude and shouldn't be off by more than 10 meters from the actual value at any given time. It need only be correct from sea-level to 4,000 feet above sea level. It is solely for use by hikers to judge altitude.

   In addition, for cost reasons, you are restricted to supplying 5V for power and 0V for ground (you can't use voltage dividers or something else and thus the V_{ref} must be 5V and ground must be 0V).

   The following A2D devices are on the market:

   - **The Bob**. This converter features an 8-bit output, no more than ± 1/8 LSB INL and can do 5000 conversions per second. It costs $0.25 per unit
   - **The Tom.** This converter features a 14-bit output, no more than ± 1/3 LSB DNL, and the ability to do 50,000 conversions per second. It costs $0.40 per unit
   - **The Mary.** This converter features a 12-bit output, no more than ± 1/2 LSB INL and can do 500 conversions per second. It costs $0.45 per unit.
   - **The Linda.** This converter features a 16-bit output, no more than ± 1/8 LSB INL and can do 2000 conversions per second. It costs $1.00 per unit.
   - **The Bob version 2.** This converter features a 16-bit output, no more than ± 1/8 LSB INL and can do 10,000 conversions per second. It costs $1.20 per unit.

Assuming that there are no other relevant differences between the converters, which, if any, of these would meet the requirements? Which would you recommend and why? Explain your reasoning. You may not use/assume oversampling techniques. Be clear about your reasoning—a correct answer with no explanation will get little if any points. **[12 points]**

4. Consider the following C function.

```
int add(int *w, int *y, int z)
{
     *y=doit(z);
     return(*y+w);
}
```

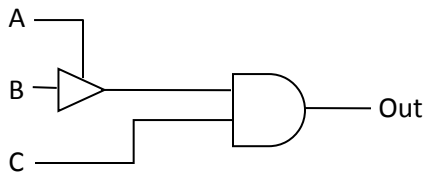
Rewrite the above code in ARM assembly for our SmartFusion while following the ABI. You are to assume the "doit()" function is ABI-compliant and has a prototype of `int doit(int)`. **[12 points]**

5. For the following program, assume you start with all memory locations in question equal to zero. Indicate the values found in these memory locations when the programs end. *Write all answers in hex.* Each memory location shown is a single byte. **[5 points, -1 per wrong box min 0]**

```
mov r2, #100
movw r1, #0x123
movt r1, #31
str r1, [r2], #3
str r1, [r2, #-1]!
strh r2,[r2, #3]
```

Address	Value (in hex)
100	
101	
102	
103	
104	
105	
106	
107	

6. Consider the circuit found below. Complete the supplied truth table. Write "HiZ" if the value is high impedance, and '?' if the value is unknown. Otherwise write a 1 or a 0 as normal. **[5, -2 per wrong or blank answer, min 0]**



A	B	C	Out
1	1	1	
0	1	1	
0	1	0	
0	0	1	

7. Consider the following code. Assume the first mov instruction is at location 0x220 and that all *instructions* are 32-bits. **[8 points]**

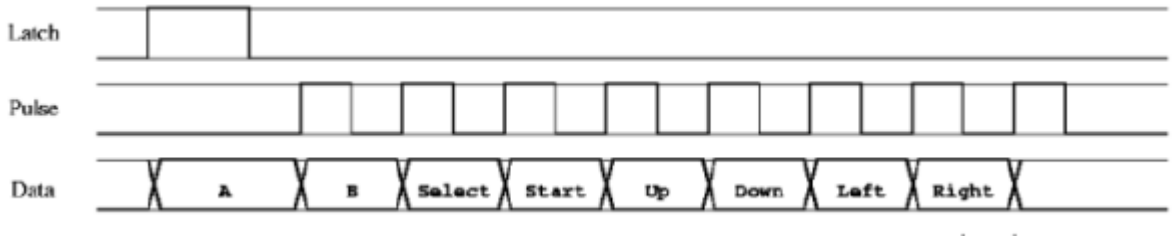
```
data:
        .byte 0x20, -6, 10, 0x0f, 16, -1, 7, 1
func:
        mov r0, #0
        mov r4, #0
        movw    r1, #:lower16:data
        movt    r1, #:upper16:data
top:    ldrb    r2, [r1],1
        add r4, r4, r2
        add r0, r0, #1
        cmp r0, #3
        bne top
done:   b done
```

- What is the value of the label "data" (in hex)? **[1]** _____
- What is the value of the label "top" (in hex)? **[1]** _____
- What is the final value of r0 (in decimal)? **[1]** _____
- What is the final value of r1 (in decimal)? **[2]** _____
- What is the final value of r4 (in decimal)? **[3]** _____

Design Problem: Nintendo 8 Interface

Your task is to design a system that will read the N8 game controller, translate the value to an ASCII value and write it to a UART. You should read the entire problem thoroughly before starting.

The game controller is essentially a shift register that loads the state of the 8 buttons on the rising edge of the "latch" signal and shifts the value out serially with subsequent "pulses" as shown below. The data is easily read by the game processor with another shift register. The data is persistent at each transition so each button value can be latched and shifted into a shift register on each rising edge of pulse. For example, the **A** button can be read on the first rising edge of Pulse, the **B** button can be read on the rising edge of the 2nd pulse….with the **Right** button being able to be read on the last rising edge of pulse.



The process is applied 60 times a second as shown in the following waveform. The controller will be read at 60Hz (so a new set of pulse/latch signals would start every 60th of a second.

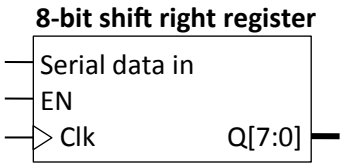
Hint/comment: In class you've repeatedly been told you generally shouldn't use anything other than the clock as a clock unless you really need to. In this case, you are dealing with a problem where you are more-or-less forced to do so. In parts of this design question you will likely need to use latch and/or pulse as a clock input.

Part 1: Nintendo 8 Module **[14 points]**

Your task is to design a memory-mapped IO interface to the N8 that will capture the button values and generate a FABINT after 6 complete reads (every 10th second). The ISR is to get the buttons' values by performing a read from memory location 0x40050000. The logic which generates the Latch and Pulse signals will be given to you (**you don't have to design it!**). You can assume Latch and Pulse are free of glitches and are synchronized to PCLK though a PLL (basically the rising edge of Latch and Pulse happen on a rising edge of PCLK). When Latch and Pulse go high they are high for at least 1 µs.

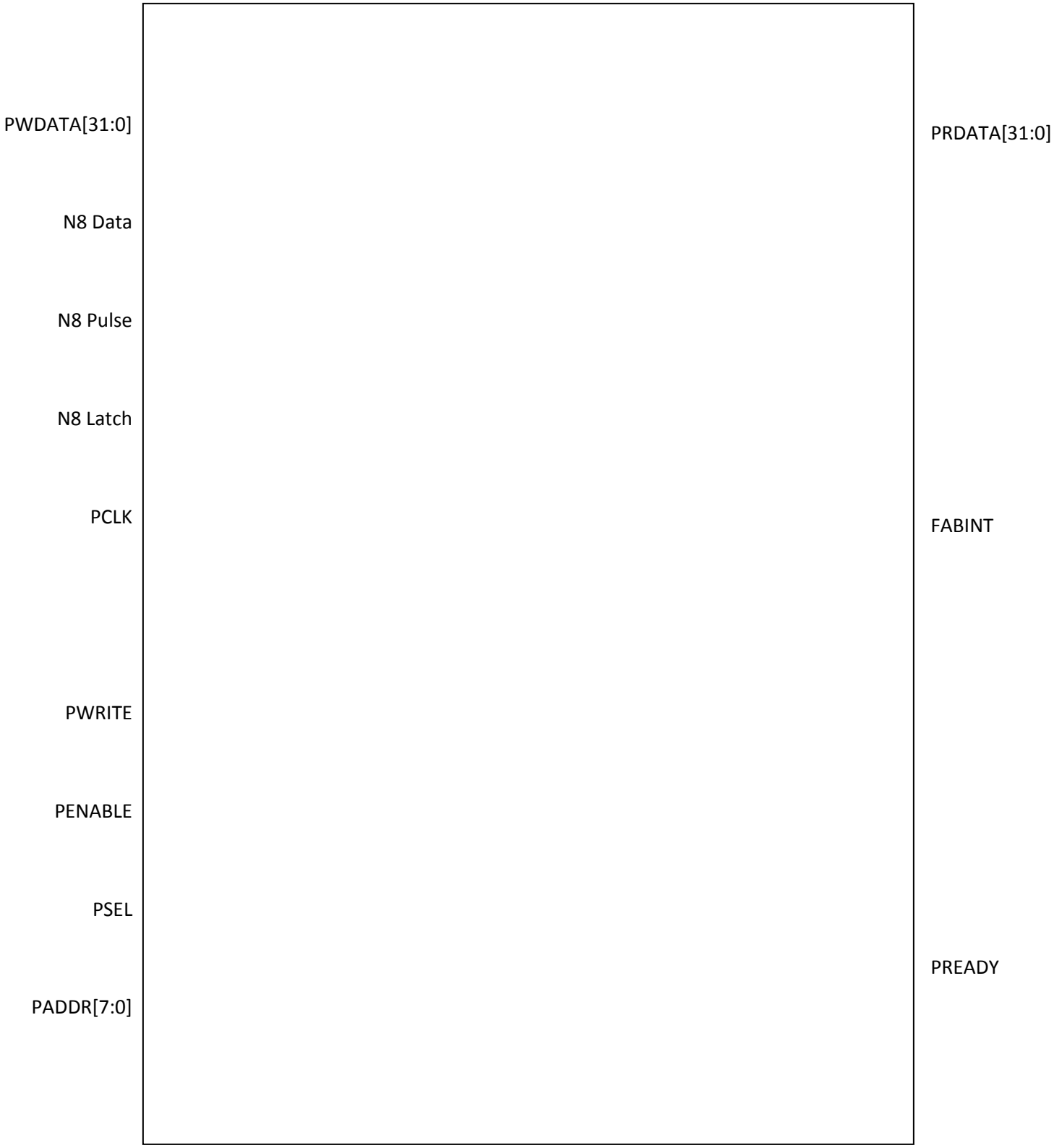
You are to design this module in schematic form. The following components are available. You may use as many of each device as you need unless otherwise indicated.

1. One serial shift register with a serial data input, an enable, and a clock input. It has an 8-bit parallel output that reflects the values in the shift register. The register shifts data to the right on every rising edge in which EN is 1.
2. One N-bit counter with synchronous reset. (You should specify the N)
3. AND, OR, XOR and NOT gates.
4. Tri-State drivers.
5. DFFs with CE (clock enable).



You may represent buses as a single wire, but indicate which signals it carries. For example, PADDR[3:0]. Further, you may show signal connections with a signal label instead of a wire. For example, you can write PCLK wherever a PCLK is needed.

Use the space on the following page to draw your schematic. The APB3 timing is provided at the end of this document. All inputs are shown on the left and all outputs on the right.


PWDATA[31:0]
N8 Data
N8 Pulse
N8 Latch
PCLK
PWRITE
PENABLE
PSEL
PADDR[7:0]
PRDATA[31:0]
FABINT
PREADY

Part 2: UART Module **[14 points]**

Your task is to develop a memory-mapped-IO interface in Verilog to a UART module provided for you. The port definition of the UART module is as follows.

```
module uart(
input baud_clk,
input [7:0]rcv_data,
input serial_data_in
output[7:0] xmt_data,
output rcv_data_rdy,
output xmt_data_rdy,
output serial_data_out);
```

- The baud clk port is used to provide a clock at 1 Mhz to generate the baud rate.
- The rcv_data port is an 8-bit value representing the received value. It is available when rcv_data_rdy is logical 1.
- The serial_data_in port is the UART's serial data input line.
- The xmt_data port is an 8-bit value used to send data. It is available when xmt_data_rdy is logical 1.
- The rcv_data_rdy and xmt_data_rdy are as described above.
- The serial_data_out port is connected the the UARTs serial output line.

Although our application only requires sending data over the UART, we want the module to work for both sending and receiving data. Instantiate the UART module and provide a MMIO interface that will read and write data and status at the following memory locations to be used in the interrupt service routine.

Value	Memory Location	data bits	Read or Write
rcv_data	0x40050100	[7:0]	Read
rcv_data_rdy	0x40050104	[0]	Read
xmt_data_rdy	0x40050108	[0]	Read
xmt_data	0x4005010C	[7:0]	Write

You may assume PCLK is 100 Mhz. As hinted at by the addresses, this module is has a different PSELECT signal than the module in Part 1.

Use the following page to write your Verilog. The module header is provided for you.

Extra space is provided on the following page.

```
module uart_interface(
        input PCLK,
        input PRESERN,
        input PSEL,
        input PENABLE,
        input PWRITE,
        input [7:0]PADDR,
        output wire PREADY,
        output wire PSLVERR,
        input [31:0] PWDATA,
        output  [31:0] PRDATA,
        serial data_out,
        serial data_in);
```

Extra Space for Verilog

Part 3: Interrupt Handler **[12 points]**

Write an interrupt handler that reads the N8 controller, converts the bit positon into an ASCII (text) string that represents the button and sends the button name over UART. For example if Select should be "Select", A should be "A", etc. *Assume that only one button is pushed at a time*. You may find it helpful to use the functions log2 (computes the log base 2 of the input) and strlen (computes the number of characters in the input string). The prototypes for those functions are: **`double log2(double x);`** and **`uint32_t  strlen(char * str);`**.

```
#include <string.h> //needed for strlen()
#include <math.h>   //needed for log2()

__attribute__ ((interrupt)) void Fabric_IRQHandler( void ) {
```

APB Bus Read Write Timing Reference

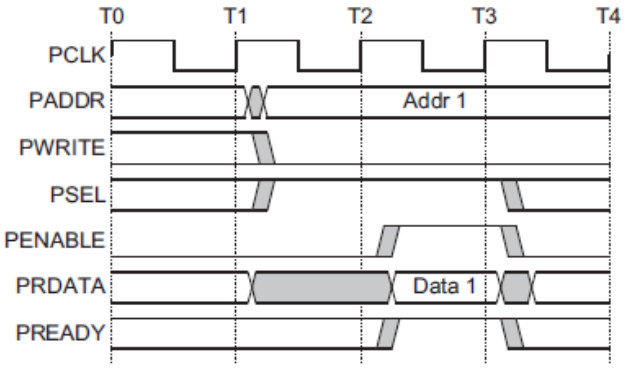


Figure 2-3 Read transfer with no wait states

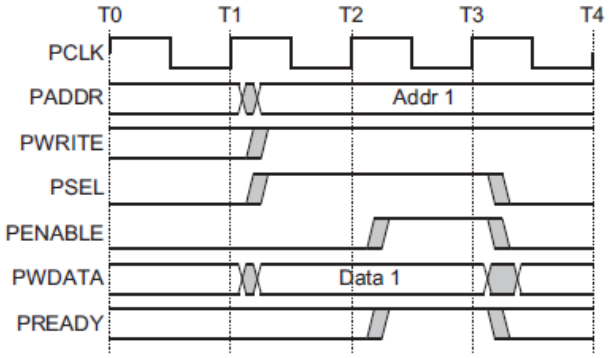


Figure 2-1 Write transfer with no wait states