EECS 373 Midterm 2
Winter 2022

24 March 2022

No calculators, reference material, or internet.

Name

UM Uniqname

Sign below to acknowledge the Engineering Honor Code: “I have neither given nor received aid on this examination, nor have I concealed a violation of the Honor Code.”

Signature

The number of points per problem are not well correlated to the time required. This is intentional, as we want students with good basic knowledge to get reasonably high scores, but for students with deeper understanding to receive higher scores. After the points per problem, we indicate our estimates of the relative amount of time required for each question. There are two “long” questions: 5 and 6.

1 Power (10 points, short time)

Use at most one sentence to indicate the main potential source of modeling error when using the “tabular method” of embedded system power modeling described in lecture?

Several answers were accepted. For example, “Changes in power consumption that depend on the combination of states of different components are not modeled;” this is due to the model using a sum of the state-dependent power consumptions of individual components. Answers focusing on the inability of low temporal resolution sampling to capture spikes in power consumption are also correct, although I view this as a less fundamental property of the modeling approach than the first answer. There were also many answers that indicated lack of exposure to the concepts in the lecture. I suggest that those who received few points on this question review the lecture video and if it remains unclear, see me in office hours. This is something you will almost certainly benefit from in your career if you design energy-constrained systems, e.g., battery-powered systems, and the model is not complex. It should take no more than 15 minutes to cover it in office hours.

2 ADCs (12 points, moderate time)

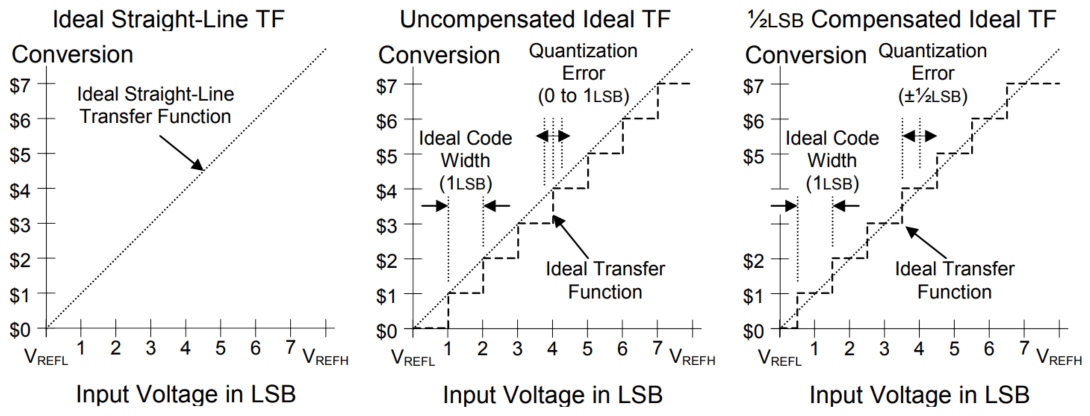


Figure 1: Quantization graphs (figure credit to Freescale).

Consider a 10-bit ADC with $V_{REF-} = -2.5\,\text{V}$ and $V_{REF+} = 2.5\,\text{V}$, whose properties are illustrated in Figure 1. The input is 1/2 LSB Compensated. Assume that it has an ideal transfer function. For this question, it is fine to give your answers in the form of fractions to reduce the amount of manual calculation necessary.

1. What is its maximum quantization error (In volts)? (6 points)

$\frac{V_{REF+} - V_{REF-}}{2^n \cdot 2}$ **where** n **is the number of bits and the additional 2 factor in the denominator comes from the 1/2LSB compensation.** $\frac{2.5\,\mathrm{V} - -2.5\,\mathrm{V}}{2^{10} \cdot 2} = 5\,\mathrm{V}/(2^{11}) = 0.002441\,\mathrm{V} = 2.441\,\mathrm{mV}$.

2. What voltage would correspond with the ADC value of 377 (decimal)? (6 points)

NOTE: We do not require a voltage range, only a nominal value with the supplied ADC count

The primary correct equation is:

$$
\begin{aligned}
V &= N_{adc} \cdot \frac{V_{REF+} - V_{REF-}}{2^n} + VREF- & (1)\\
&= \frac{377 \cdot 5\,\mathrm{V}}{2^1 0} - 2.5\,\mathrm{V} & (2)\\
&= \frac{377 \cdot 5\,\mathrm{V}}{1024} - 2.5\,\mathrm{V} & (3)\\
&= \frac{377 \cdot 5\,\mathrm{V}}{1024} - 2.5\,\mathrm{V} & (4)\\
&= 1885/1024\,\mathrm{V} - 2.5\,\mathrm{V} & (5)\\
&= -675/1024\,\mathrm{V} & (6)\\
&= -0.65917\,\mathrm{V} & (7)
\end{aligned}
$$

We will not be taking off points for the following equation which adjust the denominator by minus 1. $V = N_{adc} \cdot \frac{V_{REF+} - V_{REF-}}{2^n - 1} + V_{REF-}$

3 Timers (14 points, moderate time)

You are part of a large EECS 373 project team that decided to barely meet your difficulty point requirements by developing a fully functional robotic exoskeleton. Your role on the team is to set up timers to interface with the various servos and actuators used to control the robot. Assume the internal source clock of the microcontroller is set to 12 MHz. The values used should be for a generic timer module, not the specific values to put inside the registers of a microcontroller like the STM32L4R5ZI used in lab.

1. What prescalar is necessary for the timer to have a resolution of 5 µs? (7 points)

$$\frac{\textbf{Period desired}}{\textbf{Source clock period}} = \frac{5 \times 10^{-6}\,\text{s}}{1/(12 \times 10^{6})\,\text{s}} = 60 \textbf{(so the prescalar for an STM32 would be 59)} \tag{8}$$

2. If the timer module has 12 available bits, what is the maximum amount of time one PWM period can take? Use the prescaler from Part 1. (7 points)

1 count takes 5 µs. Maximum of 2^{12} values in the timer. 5 µs $\cdot 2^{12} = 20.48\,\text{ms}$

4 Memory (10 points, short time)

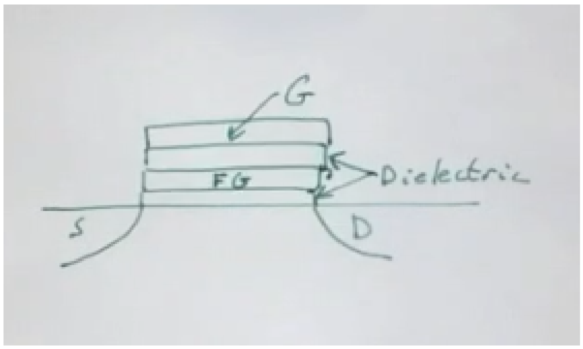


Figure 2: Floating gate transistor.

Recall the structure of a floating-gate transistor, as explained in lecture using Figure 2. Note that this is the NFET-like variant. How does one read the floating-gate memory cell?

- ◯ Set V_{GS} to a very low value, repelling electrons in the floating gate to the source and drain, and measure how much charge was removed from the source and drain as a result.
- ◯ Measure the voltage on the floating gate.
- ◯ Expose the device to ultraviolet light and measure the resulting changes in V_{GS}.
- ◯ Set V_{GS} high and measure current into the control gate.
- ◯ Set V_{GS} and V_{DS} high and observe whether current flows from drain to source.

E.

5 Serial Communication (20 points, long time)

1. For each list of signals, below, indicate the serial communication protocol using them.

   SDA, SCL (2 points): ◯ UART ◯ SPI ◯ I2C

   I2C

   RXD,TXD, RTS, CTS (2 points): ◯ UART ◯ SPI ◯ I2C

   UART

   MOSI, MISO, SS, SCLK (2 points): ◯ UART ◯ SPI ◯ I2C

   SPI

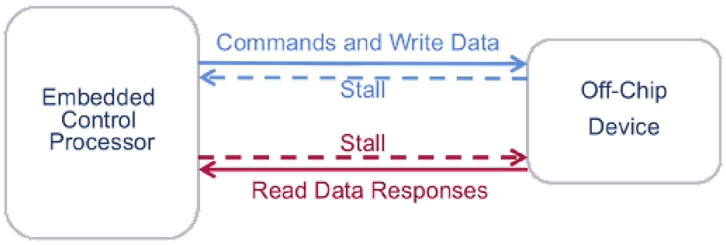


Figure 3: Communication diagram.

2. (7 points) You are designing a system that has an off-chip device that needs to have both data and commands sent to the device, while also being able to respond with data on a read request. Both the device and control processor need a mechanism to stall transactions. Figure 3 illustrates the system.

   If the device can't support the needed functionality, then write "N/A", otherwise write the number of wires each of serial interfaces would require for communication (not including power or ground).

   I2C: **2**
   SPI: **n.a.**
   UART: **4**

   Note: SPI is unable to be used here because the system requires the ability to stall the devices, and SPI does not provide this functionality. UART can provide the stalling functionality through the RTS and CTS signals.

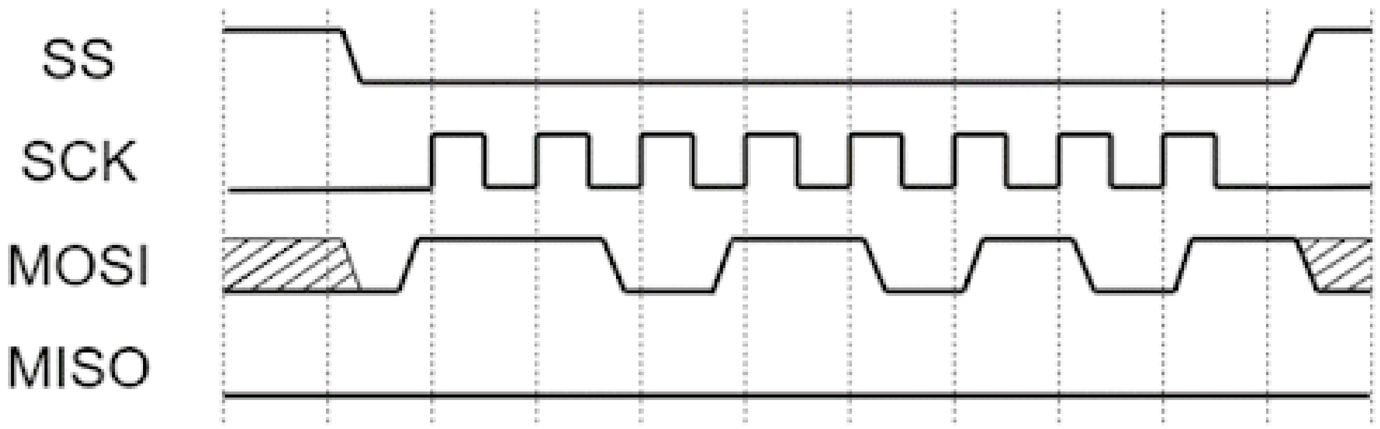


Figure 4: Timing diagram.

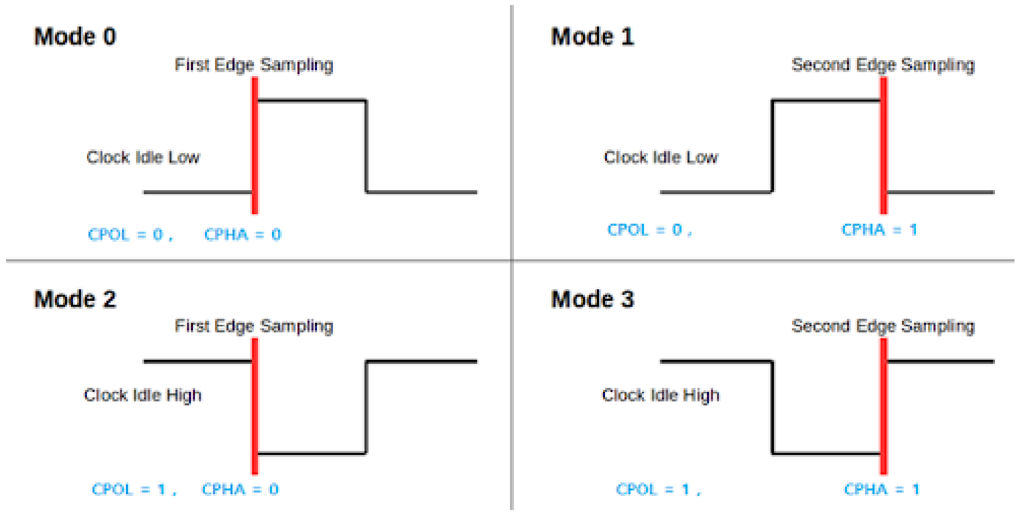


Figure 5: SPI documentation.

3. (7 points) As an embedded systems expert, you have been called in to help Team Apple debug their SPI system. The system consists of two off-chip devices connected to a master microcontroller through a shared bus and all communication is done using the SPI protocol. Unfortunately, their documentation for the system was corrupted and Team Apple is unable to identify the SPI operating mode for the off-chip devices. Luckily, the off-chip devices can display the most recent data they received, and you are able to deduce the operating mode (and consequently the clock polarity and clock phase) by examining a waveform and the devices' displayed data. Referring to Figures 4 and 5, what should the master microcontroller set its clock phase and polarity to communicate to each device?

   Data are presented MSB (i.e., the first data sample is the most significant bit).

   Device A — Data : 0b11011010 CPOL: **0**
   CPHA: **0**

   Device B — Data : 0b11010101 CPOL: **0**
   CPHA: **1**

   Note: The idle state of the clock is low, so we will have CPOL = 0 for both devices. Then we can inspect the rising and falling edges relative to MOSI to determine each device's CPHA value.

6 APB (24 points, long time)

Figure 6: Buzz Lightyear stuck with a bunch of green alinens in a square arcade claw machine.

Figure 6 illustrates the situation. Your job is to design an automated claw to seek and and capture Buzz.

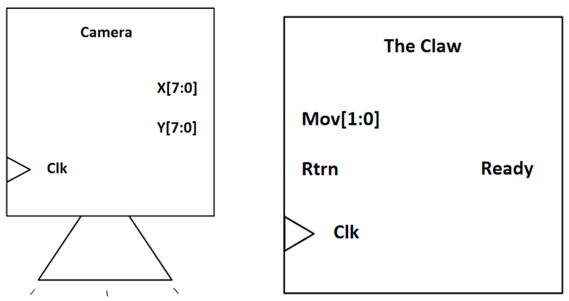


Figure 7: Camera and claw diagrams.

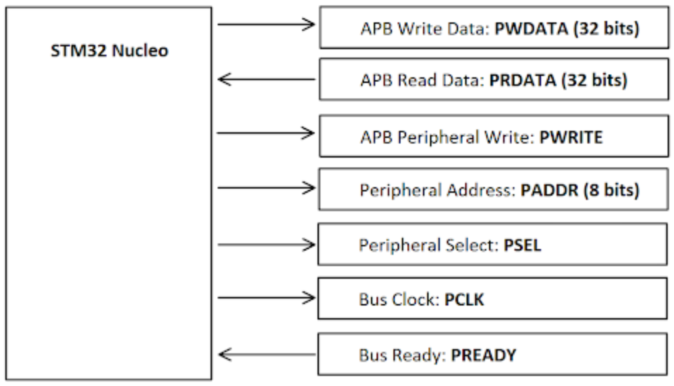

STM32 Nucleo
APB Write Data: **PWDATA (32 bits)**
APB Read Data: **PRDATA (32 bits)**
APB Peripheral Write: **PWRITE**
Peripheral Address: **PADDR (8 bits)**
Peripheral Select: **PSEL**
Bus Clock: **PCLK**
Bus Ready: **PREADY**

Figure 8: Nucleo diagram.

Hardware: You are given two devices that you will attach to the APB Bus in order to communicate with your STM Board. The first is a camera that sends the (x,y) coordinates of Buzz. The second is a claw that moves in increments and can grab whatever is underneath it and then return it to the (0,0) home position.

Camera: The camera shown in Figure 7 outputs two 8 bit values (X[7:0] and Y[7:0]) that represent the location of Buzz in the square tank. (0,0) represents the bottom left corner and (255,255) represents the top right corner of the tank. The X and Y values update with a slight delay for each rising edge of the ClK input.

Claw: Figure 7 illustrates the claw. Inputs

1. Mov[1:0]: Tells the claw which direction to go - up (0,0), down (0,1), left (1,0) or right (1,1) are the commands. Each move cycle moves by 1/256th of the length or width of the tub, and therefore one camera pixel at a time. The claw cannot move outside the region of the camera.
2. Rtrn: If a 1, The claw grabs whatever it is over, and then returns to the home position (0,0) with that object. If a 0, a move action will occur instead.
3. Clk: Actions will execute at the rising edge of the clock,

Ready Output: Outputs a 0 when the Claw is not done completing its action. Outputs a 1 when the Claw is done completing its previous action and is ready to receive the next Clk command.

Interfacing the APB Bus: The kit has the APB bus interface shown in Figure 8. The signal names are shown in bold. Read and write cycles are provided on the bottom of the page. PSEL is configured to be “1” when memory locations 0x40002000-0x4000200F are accessed.

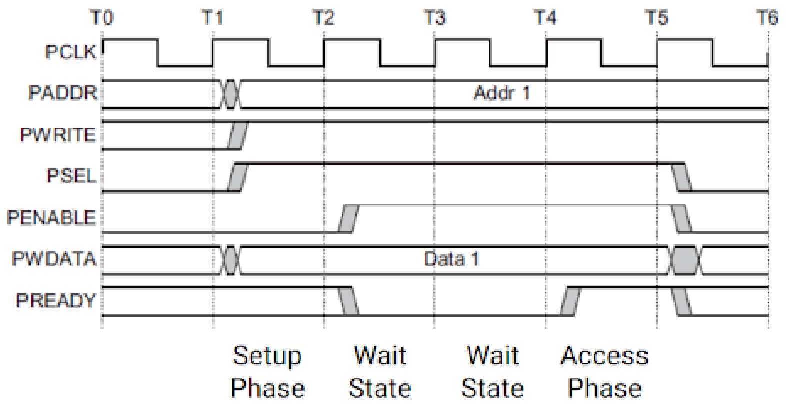


Figure 9: Write transfer with a wait state.

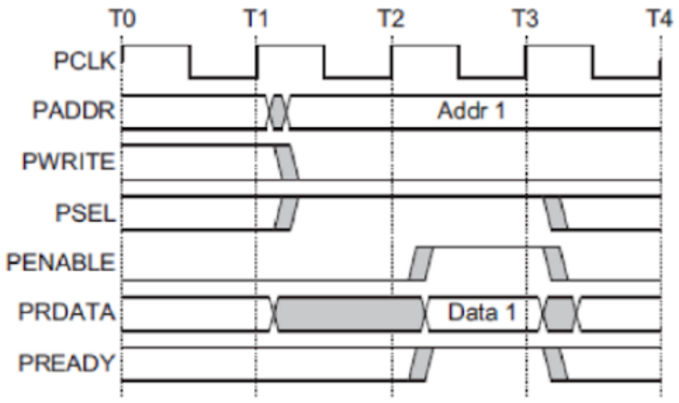


Figure 10: Read transfer without a wait state.

Part 1: Hardware (12 points)

Use Figure 11 to design an APB-compliant hardware module by connecting the peripherals in Figure 7 with wires and standard logic gates, such that the module will give an up to date location of Buzz Lightyear from the camera and be able to maneuver the claw.

1. A read from location 0x40002000 should read Buzz's location. X[7:0] from data bits [7:0], and Y[7:0] from data bits [15:8].
2. A write to location 0x40002004 will maneuver the claw. Data bits [1:0] correspond to the Mov[1:0] bits sent to the claw, and data bit [2] represents the value of Rtrn.

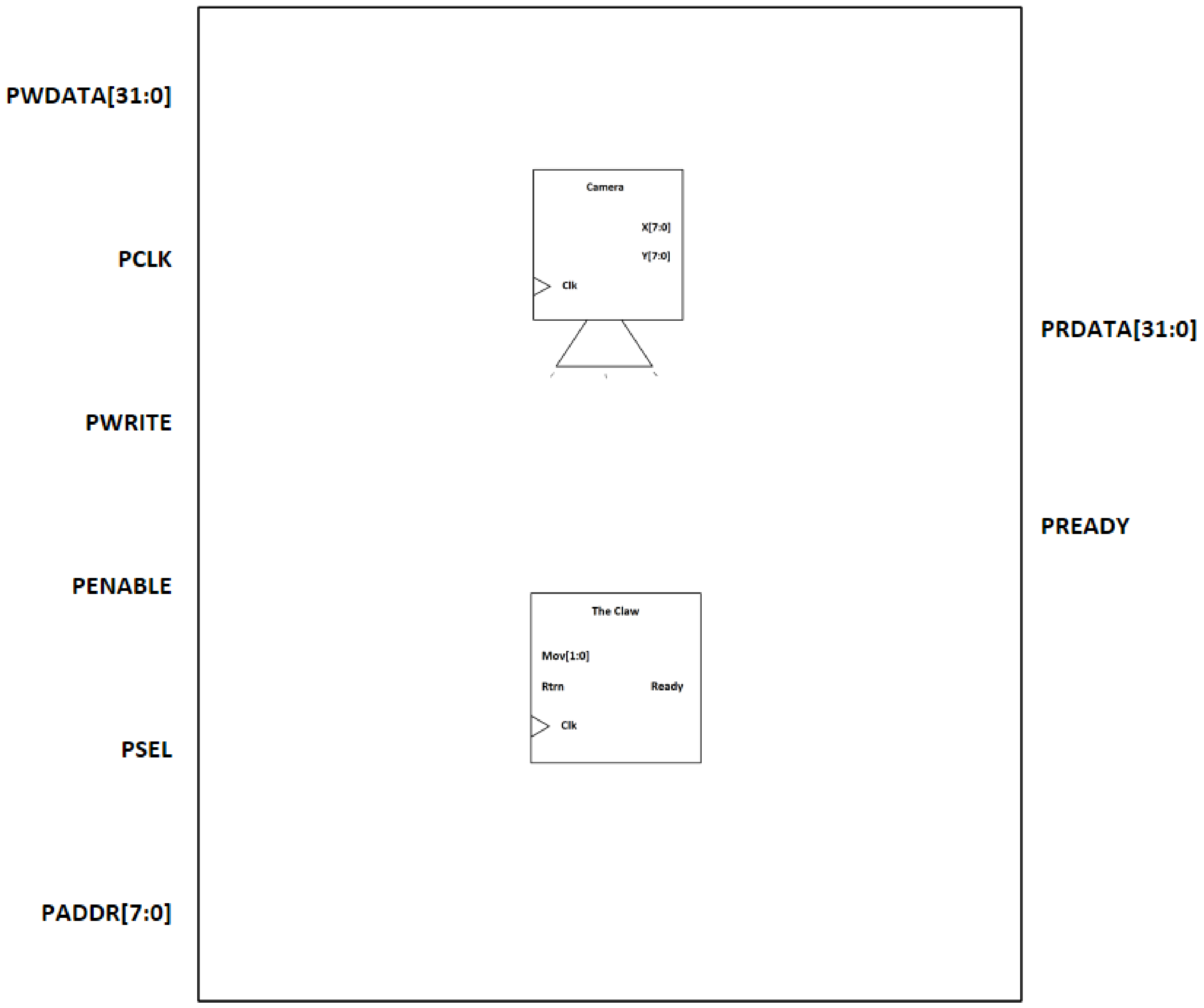
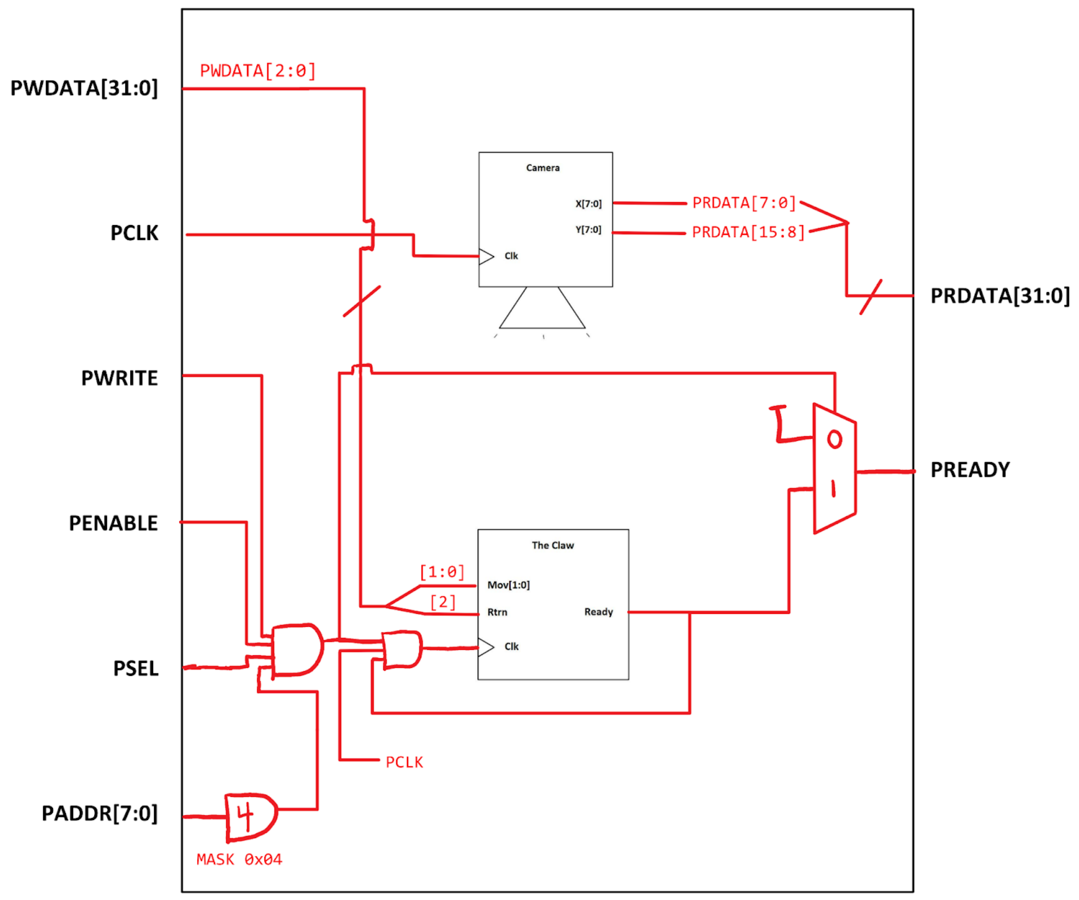


Figure 11: System diagram and answer box.


PWDATA[31:0]
PWDATA[2:0]
PCLK
Camera
X[7:0]
Y[7:0]
Clk
PRDATA[7:0]
PRDATA[15:8]
PRDATA[31:0]
PWRITE
PENABLE
PSEL
PADDR[7:0]
1
0
1
PREADY
The Claw
[1:0]
[2]
Mov[1:0]
Rtrn
Ready
Clk
PCLK
4
MASK 0x04

Part 2: Software (12 points)

Your task is to write a function in C that autonomously tracks down Buzz Lightyear.

- Assume that the claw starts at position (0,0).
- Also note that Buzz Lightyear moves around and will need to be tracked down.
- The function should end when the Rtrn action occurs and Buzz and the Claw are in the same location, i.e., when Buzz is caught.
- Despite his name, Buzz Lightyear is not as fast as the claw and is catchable.
- If you use an infinite loop to implement the function, you are welcome to call the `wait_for_change_delay()` function, which idles the processor until sensor inputs change, thus converting a processor-hogging infinite loop into an event-driven scheme.

```
void Catch_Buzz(void) {

    volatile uint16_t * camera_addr = (uint16_t *)0x40002000;
    volatile uint8_t * claw_addr = (uint8_t *)0x40002004;
    uint8_t up = 0b00, down = 0b01, left = 0b10, right = 0b11, rtn = 0b100;
    uint8_t clawPstn[2] = {0,0}; uint8_t buzzPstn[2];
    uint8_t buzzCaught = 0;

    while (!buzzCaught) {
        *((uint16_t *)buzzPstn) = *camera_addr; // Assume little endian
        if (clawPstn[0] < buzzPstn[0]) {
        *claw_addr = right;
        ++clawPstn[0];
    } else if (clawPstn[0] > buzzPstn[0]) {
        *claw_addr = left;              –clawPstn[0];
    } else if (clawPstn[1] < buzzPstn[1]) {
        *claw_addr = up;
        ++clawPstn[1];
    } else if(clawPstn[1] > buzzPstn[1]) {
        *claw_addr = down;
        –clawPstn[1];
    } else {
        *claw_addr = rtn;
        buzzCaught = 1;
    }
    return;
}
```

7 PCB design (10 points, short time)

Indicate the PCB components associated with the following (somewhat cryptic) definitions.

Electronic devices that sit atop the PCB (2 point).

◯ traces ◯ vias ◯ surface-mount ◯ through-hole ◯ ground/power plane

surface-mount

Copper tracks on the surface of the PCB (2 point).

◯ traces ◯ vias ◯ surface-mount ◯ through-hole ◯ ground/power plane

traces

A copper layer almost entirely covering a layer of the PCB (2 point).

◯ traces ◯ vias ◯ surface-mount ◯ through-hole ◯ ground/power plane

ground/power plane

Small copper-plated holes through the layers of the PCB (2 point).

◯ traces ◯ vias ◯ surface-mount ◯ through-hole ◯ ground/power plane

vias

Electronic devices whose pins penetrate holes in the PCB (2 point).

◯ traces ◯ vias ◯ surface-mount ◯ through-hole ◯ ground/power plane

through-hole