Project 4

Due Sept 9, 2025 at 9:00 PM

This project description is subject to change at any time for clarification. For this project you will be working with a partner.

Desired Outcomes

- Exposure to using circuit simulator (Logisim Evolution)
- Exposure to interrupt/system call mechanism
- An understanding of how carry-look ahead is implemented
- An understanding of how to develop a simple RISC style CPU

Project Description

You will be using Logisim Evolution Version 3.9.0 for this project. You may use any of the built-in components of Logisim Evolution. All class projects will be run through MOSS like software to determine if students have excessively collaborated. Excessive collaboration, or failure to list external sources will result in the matter being referred to Student Judicial Affairs

You will be developing a Variable Pulse Width Bus Controller (VPWBC). You will need to at least be able to send/receive frame correctly to receive full credit. The support for the interrupt line, and frame filtering will be extra credit. The VPWBC you will be developing will be capable of sending and receiving 12-bit frames on a shared bus. The VPWBC will have a 12-bit parallel interface to communicate with the micro-processor. The micro-processor side parallel interface includes a $\overline{CS}$, $\overline{O}$, $\overline{W}$, $\overline{INT}$, 1 dedicated address lines ADDR, the 12 data in lines DATAIN, and the 12 data out lines DATAOUT. The other I/O include a CLK, SER_IN, and SER_OUT. The VPWBC has a 4 frame TX FIFO and a 4 frame RX FIFO.

Registers

Status/Control (Address 0)

The register accessed at address 0 depends upon transaction type (read or write).

	11	10	9	8	7	6	5	4	3	2	1	0
Read	C_3	C_2	C_1	C_0	A_3	A_2	A_1	A_0	TXI	TBNF	DR	ENA
Write	C_3	C_2	C_1	C_0	A_3	A_2	A_1	A_0	TXIIE	TBNFIE	DRIE	ENA

C_3 – C_0 – Care bits for receive frames. If the bit C_X is set, then the corresponding address bit A_X must match the received frame otherwise the frame is discarded.

`A_3 – A_0 – Address bits for receive frames. See care bits above.

TXI – Transmitter Idle, this bit is set if the transmitter is not outputting anything. This bit is cleared when the transmitter is shifting out data.

TXIIE – Transmitter Idle Interrupt Enable, this bit controls if an interrupt is generated in the case that the Transmitter is Idle.

TBNF – Transmit Buffer Not Full, this bit is set if the transmit buffer is not full. This bit is cleared if when the TX FIFO becomes full.

TBNFIE – Transmit Buffer Not Full Interrupt Enable, this bit controls if an interrupt is generated in the case that the Transmit Buffer is not full.

DR – Data Ready, this bit is set if there is data in the RX FIFO. This bit is cleared if all the data has been read out of the RX FIFO.

DRIE – Data Ready Interrupt Enable, this bit controls if an interrupt is generated in the case that data is ready.

ENA – Enable, this bit is set if the VPWBC is enabled, it is cleared if the VPWBC is disabled. This bit can be written at any time.

Data (Address 1)

The register access at address 1 depends upon transaction type (read or write). Upon read the next valid data frame will be read from the RX FIFO. Writing to address 1 will place a frame into the TX FIFO unless the FIFO is full.

Parallel Interface Timing

The parallel interface is asynchronous to the VPWBC clock, and each transaction (read or write) is guaranteed to last longer than one clock cycle. The a $\overline{\mathrm{CS}}$ line will go low at or prior to the $\overline{\mathrm{O}}$ or $\overline{\mathrm{W}}$ signals going low. The data must be made available for reads on the following clock edge of the VPWBC. Figure 1 shows the timing diagram for a read where DATA is the DATAOUT from the VPWBC, and Figure 2 shows the timing diagram for a write where DATA is the DATAIN to the VPWBC.

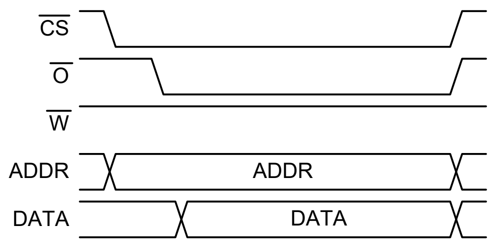

Figure 1. Data Read

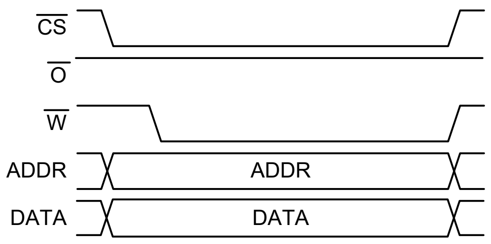

Figure 2. Data Write

Serial Interface

Each data frame is sent most significant bit first from address to End of Frame bit. Table 1 shows the VPWBC frame format. When the TX is idle the SER_OUT outputs a logic 0. Data bits are encoded in a Variable Pulse Width where the bus will transition between 0 and 1 or 1 and 0 for each subsequent bit. During transmission if the data value matches the bus logic value, then the pulse width is short (1 cycle), if it does not match then the pulse width is long (2 cycles). Figure 3 shows an example of the bit encoding for the bus, it starts with three 0 bits signifying bus idle, followed by the value 100111. The VPWBC uses a pull-up to high for the recessive state (logic zero), and a pull-down to low for the active state (logic one). The physical transmission of bits on the VPWB allows for a non-destructive arbitration.

Table 1. VPWBC Frame Format.

Bit Location	Description
0..3	Address
4..11	Data Byte
12	End of Frame (1)

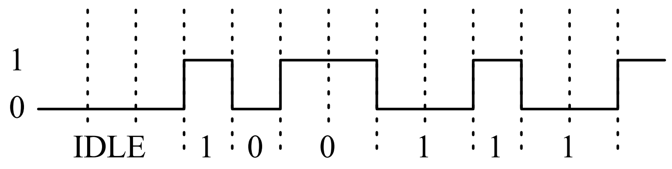

Figure 3. VPW Bit Encoding

You can unzip the given tgz file with utilities on your local machine, or if you upload the file to the CSIF, you can unzip it with the command:
```
tar -xzvf proj4given.tgz
```

You **must** submit the circuit file, README.md file, and `.git` directory in a tgz archive. You can tar gzip a directory with the command:
```
tar -zcvf archive-name.tgz directory-name
```

You should avoid using existing circuits as a primer that are currently available on the Internet. You **MUST** specify in your README.md file any sources of circuits that you have viewed to help you complete this project. Your README file **MUST** have both partner's name and SID number, a brief description of what circuits work/don't work, and a list of sources you used for designing of your circuit (you do not need to list the book or lecture notes it is assumed these have been used). You **MUST** properly document **ALL** uses of Generative AI following the guidelines outlined in the Generative AI Restrictions. All class projects will be submitted to MOSS like software to determine if students have excessively collaborated. Excessive collaboration, or failure to list external code sources will result in the matter being referred to Student Judicial Affairs.

Grading

The point breakdown can be seen in the table below. Make sure your circuit executes correctly in Logisim Evolution 3.8.0 as that is where it is expected to execute. You will make an interactive grading appointment to have your assignment graded. You must have a working webcam for the interactive grading appointment. Project submissions received 24hr prior to the due date/time will received 10% extra credit. The extra credit bonus will drop off at a rate of 0.5% per hour after that, with no additional credit being received for submissions within 4hr of the due date/time.

Points	Description
10	Has git repository with appropriate number of commits
10	Has README.md with proper documentation
10	VPWBObserver implemented as specified
10	VPWBArbiter implemented as specified
5	VPWBSRO implemented as specified
5	VPWBSRI implemented as specified
5	VPWBFrameQueue implemented as specified
10	VPWBRX implemented as specified
10	VPWBTX implemented as specified
10	VPWBController implemented as specified
5	VPWB operates as specified
10*	Student understands all circuits they have provided
Extra Credit	
25	Extra credit implemented as specified.

* Students who are unable to demonstrate understanding of their circuit could receive negative points and resulting in score as low as zero overall regardless of functioning of circuit submitted.

** Groups where partner workload is not balanced may have adjustments in their scores.

Extra Credit

Implementation of the interrupt and frame filtering is extra credit. The interrupt line is lowered when an event occurs, and the associated interrupt enable bit is set in the control register. The address and care bits in the status/control register specify a filtering for received frames. If the received frame address ANDed with the care bits matches the address in the status/control register ANDed with the care bits then the frame is inserted into the RX FIFO, otherwise it is discarded.

Suggested Approach

Many submodules have been provided to simplify the implementation of the VPWBC. The suggested order of implementation is VPWBObserver, VPWBArbiter, VPWBSRO, VPWBSRI, VPWBFrameQueue, VPWBRX, VPWBTX, and finally VPWBController. The submodules are described below.

VPWBObserver – This module observes the status of the bus and determines if the bus is IDLE, or if a new data bit is received. The module should be implemented such that it is in one of eight states: IDLE (after 3 or more zeros are received), B0 (one zero has been received since a one), B00 (two zeros have been received since a one), B1 (one, one has been received since a zero), B11 (two ones have been received since a zero), E1 (three ones have been received in a row, or a one has been received from another E error state), E0 (one zero has been received from E1), E00 (two zeros have been received from E1). Once the module enters the error state E1 it will remain in an error state until an IDLE is detected; therefore, no new data bits will be detected while in an error state. The I/O for the module are:

- RX – the input from the VPWB
- CLK – the clock
- ENA – module enable, the module is disabled if ENA is false
- IDLE – outputs true if the bus has been detected to be in the IDLE state
- DIN – the value of the new data bit
- NEWBIT – true if a new data bit was detected, false otherwise

VPWBArbiter – This module is responsible for transmitting bits and detecting if arbitration has been lost. This module should implement three states: IDLE (the transmitter is not trying to send any data bits), BIT1 (the first bit of a possible two-bit sequence is being transmitted), BIT2 (the second bit of the long sequence is being transmitted). If the transmitted bit does not match the received during BIT1 or BIT2 states, then arbitration has been lost. The I/O for the module are:

- RX – the input from the VPWB
- TX – the output to the VPWB
- CLK – the clock
- ENA – module enable, the module is disabled if ENA if false
- EVENBIT – input that specifies if the bit being written is an even bit of the frame (this means it is transmitted as a logic 1)
- DOUT – input that specifies the data bit to transmit
- DW – input that specifies that a new data bit should be transmitted
- ARBL – output that specifies that arbitration was lost while trying to send a data bit
- BITCMP – output that specifies that the bit that was being transmitted has completed, and another bit can be written

VPWBSRO – This module is responsible for shifting out a frame one bit at a time but allows for starting over on the frame if necessary. The frame is loaded into the output shift register via the PARA_IN when LOAD is true. The I/O for the module are:

- SER_OUT – output specifying that the next bit to output of the frame
- SHIFT – input specifying that the next bit should be transitioned to
- RELOAD – input specifying that the frame should be reloaded
- BITSLEFT – output specifying the number of bits left in the current frame
- CLK – the clock
- ENA – module enable, the module is disabled if ENA if false
- LOAD – input specifying that a new value should be loaded via the PARA_IN

- EMPTY – output specifying that the shift register is empty
- PARA_IN – input specifying the new frame to load

VPWBSRI – This module is responsible for shifting in a frame one bit at a time but allows for starting over if the partial frame should be discarded. The frame is loaded into the input shift register via the SER_IN and can be read out the PARA_OUT when READ is true. The I/O for the module are:

- SER_IN – input specifying the next received bit
- NEWBIT – input specifying that a new data bit has been received and should be shifted in
- CLEAR – input specifying that the partial frame should be discarded
- CLK – the clock
- ENA – module enable, the module is disabled if ENA if false
- PARA_OUT – output specifying the new received frame
- READ – input specifying that the frame should be read out PARA_OUT
- FULL – output specifying that the shift register is full and a frame can be read out

VPWBFrameQueue – This module is responsible for queuing frames. This will be used for both the TX and RX FIFOs. The I/O for the module are:

- FRAME_OUT – output specifying the next frame in the queue (head of the queue)
- DEQUEUE – input specifying that the next frame should be removed from the queue
- EMPTY – output specifying that the queue is empty
- CLK – the clock
- ENA – module enable, the module is disabled if ENA if false
- FRAME_IN – input specifying the next frame to input into the queue
- ENQUEUE – input specifying that the frame at FRAME_IN should be enqueued
- FULL – output specifying that the queue is full

VPWBRX – This module is responsible for receiving frames. This module will use an VPWBObserver, VPWBSRI, and a VPWBFrameQueue. If the other submodules have been implemented properly, then only a little glue logic should be necessary to implement this module. The I/O for the module are:

- SER_IN – input specifying the VPWB input
- CLK – the clock
- ENA – module enable, the module is disabled if ENA if false
- FRAME_OUT – output specifying the next frame in the queue (head of the queue)
- READ – input specifying that the next frame should be read out from the FIFO
- FULL – output specifying that the RX FIFO is full
- EMPTY – output specifying that the RX FIFO is empty
- CARE – input specifying the care bits from the status/control register
- ADDR – input specifying the address bits from the status/control register

VPWBTX – This module is responsible for transmitting frames. This module will use an VPWBObserver, VPWBArbiter, VPWBSRO, and a VPWBFrameQueue. This module will be a little more complicated than, the VPWBRX, but if the other submodules have been implemented properly, it should be straightforward. If arbitration is lost during the transmission of a frame, then the module must wait until the bus is IDLE before reattempting to transmit. The I/O for the module are:

- SER_OUT – output specifying the VPWB output
- SER_IN – input specifying the VPWB input
- CLK – the clock
- ENA – module enable, the module is disabled if ENA if false
- FRAME_IN – input specifying the next frame to enqueue
- WRITE – input specifying that the next frame should be put into the TX FIFO
- FULL – output specifying that the TX FIFO is full
- IDLE – output specifying that the transmitter is idle and the TX FIFO is empty