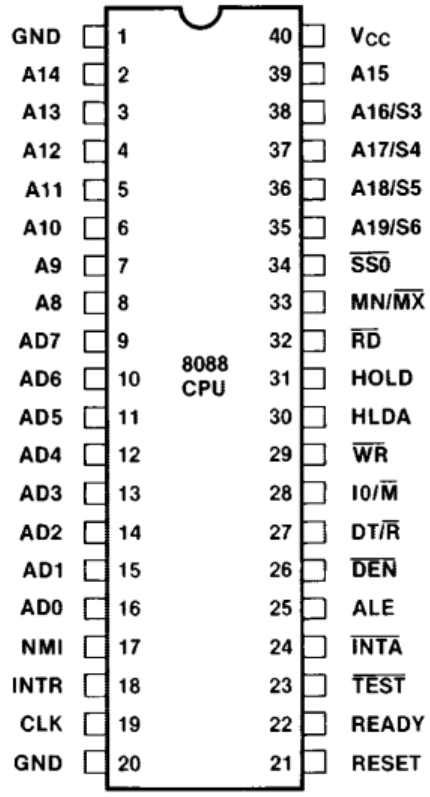
ECE 571
Introduction to SystemVerilog
Final Project Description
Winter 2024

The Intel 8088 was used in the first PC as well as countless automation, embedded systems, and industrial control applications. Today's Intel x86 processors implement extensions of the same architecture and can execute the same (binary) instructions as this early microprocessor.

The Intel 8088 Datasheet can be found here:
http://datasheets.chipdb.org/Intel/x86/808x/datashts/8088/231456-006.pdf

The 8088 could be used in so-called minimum mode or maximum mode (with a co-processor). The pinout of the chip when run in minimum mode is shown below.



This final project involves system-level design and integration, creation and use of SystemVerilog interfaces, $readmemh, bus-functional models, FSM modeling, and using protected SystemVerilog IP. To help you maintain steady progress there will be regular checkpoints.

I've created a non-synthesizable "bus functional" model of the 8088 solely for the purpose of creating bus transactions. It does not simulate 8080 or its instruction set. It is simply capable of generating read and write operations for I/O and memory. While the model is not otherwise

doing anything an 8088 would do, from anyone observing only the bus, it would appear to be a functioning (if possibly erratic) 8088.

If you need more background you can consult any of the many references for the Intel 8088 and the early PC bus. The PSU Library has several hard copies of these (some are earlier editions which are completely suitable). You may even be able to find an electronic version on-line without too much difficulty. The Rafiquzzaman book is available as an e-book through the PSU library on-line when VPNed.

Brey, B. B. (2009). *The Intel microprocessors : 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro Processor, Pentium II, Pentium III, Pentium 4, and Core2 with 64-bit Extensions: Architecture, Programming, and Interfacing* (8th ed.). Prentice Hall. Some edition of this *may* be available as an e-book though the PSU library.

Hall, D.V. (1992). *Microprocessors and Interfacing: Programming and Hardware* (2nd ed.). McGraw-Hill.

Mazidi, M.A. and Mazidi, J.G. (2003). *The 80x86 IBM PC and Compatible Computer (Volumes I & II): Assembly Language, Design, and Interfacing* (4th ed.). Prentice-Hall.

Rafiquzzaman, Mohamed. (2005). *Fundamentals of Digital Logic and Microcomputer Design* (5th ed.). J. Wiley & Sons.

A brief summary of the 8088 pins is shown below.

Signal	I/O	Function
AD7-AD0	I/O	Multiplexed data and least significant 8 bits of address
A19-A8	O	Most significant 12 bits of address (you can ignore the multiplexed use of A19-A16 as S6-S3).
CLK	I	Clock
HOLD	I	DMA request to hold the processor. When asserted, processor stops suspends execution, places address, data, and control signals in high-impedance. Should be kept at 0.
HLDA	O	Acknowledges processor has entered hold state
IO/$\overline{M}$	O	When low, indicates address is a memory address; when high, indicates I/O (port) address
$\overline{WR}$	O	When low, indicates a bus write operation
$\overline{RD}$	O	When low, indicates a bus read operation
$\overline{SSO}$	O	Used in conjunction with other signals to determine type of bus cycle
READY	I	When low, processor enters wait states; When high has no effect. Your model does not need to model wait states.
RESET	I	When asserted for four or more clock periods, resets the processor. (A real process then begins executing instructions at 0xFFFF0.)

NMI	I	Non-maskable interrupt. Should always be 0 for this assignment.
INTR	I	Interrupt request. Should always be low for this assignment.
$\overline{\text{INTA}}$	O	Interrupt Acknowledge. Will always be high for this assignment.
ALE	O	Address latch enable – indicates address/data bus contains valid address. Is not floated during a hold state
DT/$\overline{\text{R}}$	O	Data transmit/receive. When high indicates processor is putting data on the data bus (AD7-AD0). When low, indicates process is expecting to receive data from the data bus.
$\overline{\text{DEN}}$	O	Data bus enable – used to activate external data buffers.
MN/$\overline{\text{MX}}$	I	When high, indicates minimum mode. Should always be high for this assignment.
$\overline{\text{TEST}}$	I	Should always be high for this assignment

Upon being successfully reset, my model reads a file called **busops.txt**. The file format is:

```
time  type  operation  address
```

Where **time** is the elapsed time in CPU clocks, **type** is either I or M to denote an I/O or memory operation, **operation** is R or W to denote read or write, and **address** is a 20-bit address (in hexadecimal). The fields are separated by tab characters.
For example:

```
136   M     R     0x7F30B
```

Specifies that the processor should perform a memory read operation from location 0x7F30B at clock cycle 136 (or as soon thereafter if the bus is busy at that time).

When it has completed all bus transactions it calls **$finish**. If you want to do anything before simulation exits, you should create a **final** block.

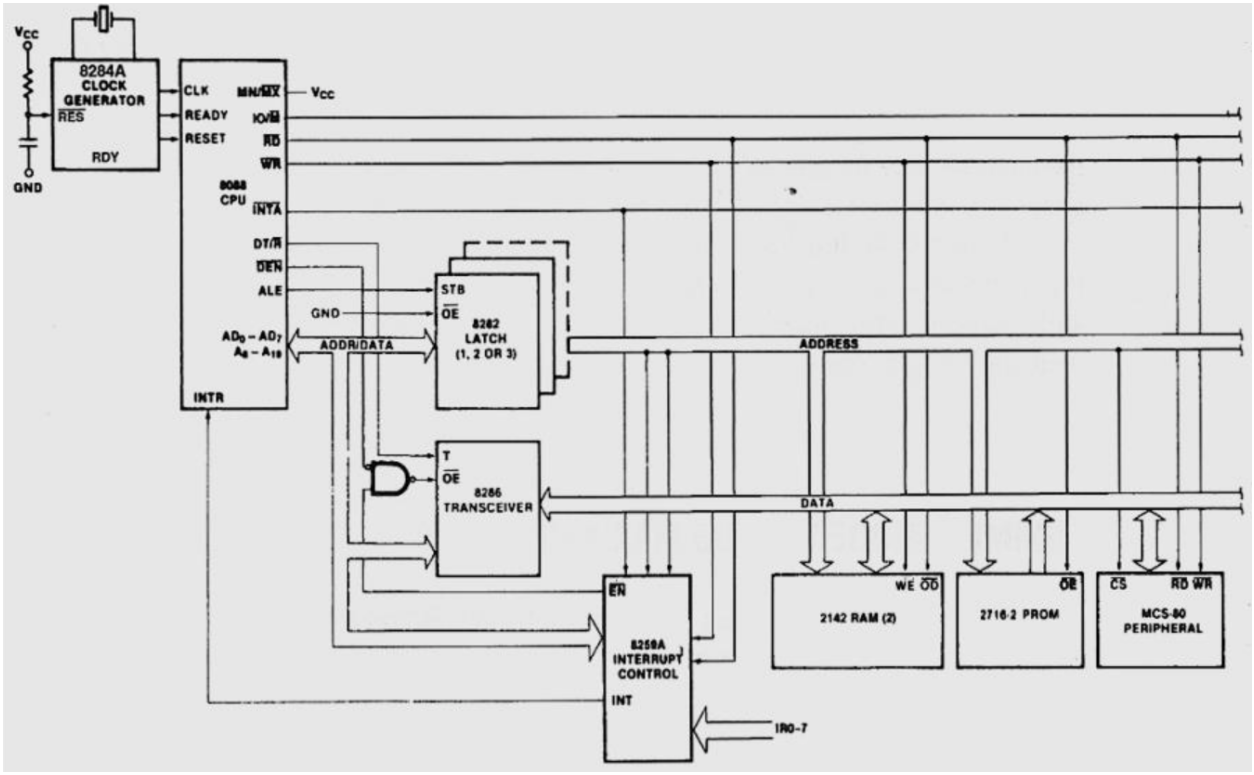
You can see how to instantiate my Intel 8088 model and how to perform the reset sequence by downloading the example top level module from the Final Project assignment page in Canvas. Note that the 8088 has a very specific reset sequence – do not alter this.

Checkpoint 0
Create a team. Familiarize yourself with the 8088 and the 8088 bus protocol and timing. This may require some outside reading. Download my Intel8088 model and run it with some different bus operations (you can create/edit your own **busops.txt** file) and observe the waveforms so you're familiar with memory and I/O reads and writes and their timing. We do not adhere to exact timing – but signals are active on the relevant clock edges.

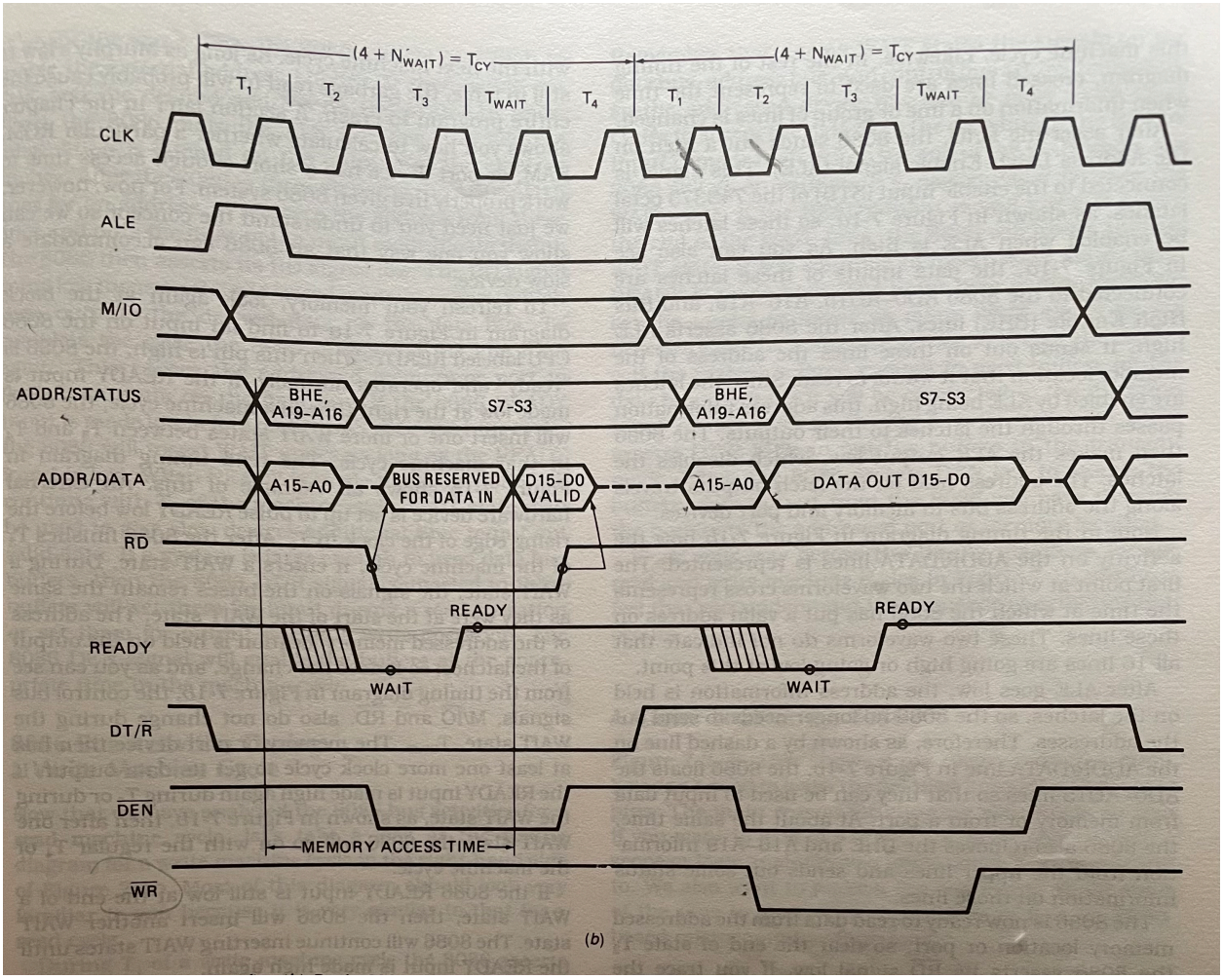
Checkpoint 1

Design a synthesizable FSM that can be used as an 8088 bus compatible memory or I/O module. It should have CS (chip select) and (OE) output enable inputs as well as ports that allow it to connect to an 8088 bus. You should use a subset of 8088 interface pins and the Address and Data signals (the output of the latch and the peripheral side of the transceiver) as created in the top level module I provided. In general, the 8088 causes output signal transitions in response to the *falling* clock edge. Thus, your IO and memory modules should probably sample them on the *rising* clock edge. Code your FSM as a one-hot Moore machine using best SystemVerilog practices. See if you can create a module can be used as a memory or I/O module depending upon how it is instantiated.

The diagram below may help you. Note that we are not modeling the clock generator or the interrupt controller.



The diagram below, taken from Prof. Hall's book, illustrates both a read and write bus operation (though you should ignore the T_{WAIT} states depicted.



Checkpoint 2

Modify the top level module to instantiate two 512KiB memories, one which responds to addresses between 0 and 512Ki-1 and the other which responds to addresses between 512Ki and 1M-1. Instantiate two I/O devices, one of which responds to port numbers 0xFF00 through 0xFF0F and the other to 0x1C00 through 0x1DFF. Modify the **`busops.txt`** file to read and write to locations in each of these four devices. Though it is not synthesizable add an **`initial`** block to the module with a filename parameter that allows the module to use **`$readmem`** to initialize the contents of the memory and state of the I/O registers.

Checkpoint 3

Create an **`interface`** for the 8088 pins with a Processor and a Peripheral **`modport`**. Make the signal names be the same as those you saw when instantiating my 8088 model with one exception: the CLK and RESET signals should be ports of the interface. Modify the top level module I provided to create a version that instantiates the interface and uses the interface (with Processor **`modport`**) when instantiating my 8088 model. I'll be providing a version of my 8088 module that uses an interface.

I've written a lot of new code for this final project and I believe I've debugged it but it's possible you may find errors. If you're convinced that you've found an error, please let me know.