# Memory Decoder Design

EECS 312 Project

---

## **Project Description**:

### Introduction

A common application for a decoder is *memory addressing*. A decoder with n inputs can address $2^n$ locations in a memory array. This significantly reduces the number of data bits being sent across the system address bus.

The design of a decoder plays a significant role in the speed and power consumption of the memory. As a circuit designer, you will get an insight into the different options for designing a decoder through this project.

### Functionality

You will design a 3:8 decoder for this project. The decoder has three data inputs, an enable input and eight outputs. When enable is high, the n<sup>th</sup> output line corresponding to the input code goes high and all other outputs are zero. When enable is low, the inputs can take any value (X = don't care) while the outputs must be zero.

The truth table for the decoder is given below.

| *Enable* | *Inputs* | | | *Outputs* | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **E** | **$D_2$** | **$D_1$** | **$D_0$** | **$Q_7$** | **$Q_6$** | **$Q_5$** | **$Q_4$** | **$Q_3$** | **$Q_2$** | **$Q_1$** | **$Q_0$** |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | X | X | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## Goal

The goal of this project is to minimize the energy*delay product subject to a delay constraint of 800ps (worst-case). This worst-case delay is measured from 50% of the input transition to 50% of the output transition. You will need to determine the longest path in your circuit and the worst-case input patterns.

## Teams

This is a group project to be done in teams of two. You are free to choose your own teams.

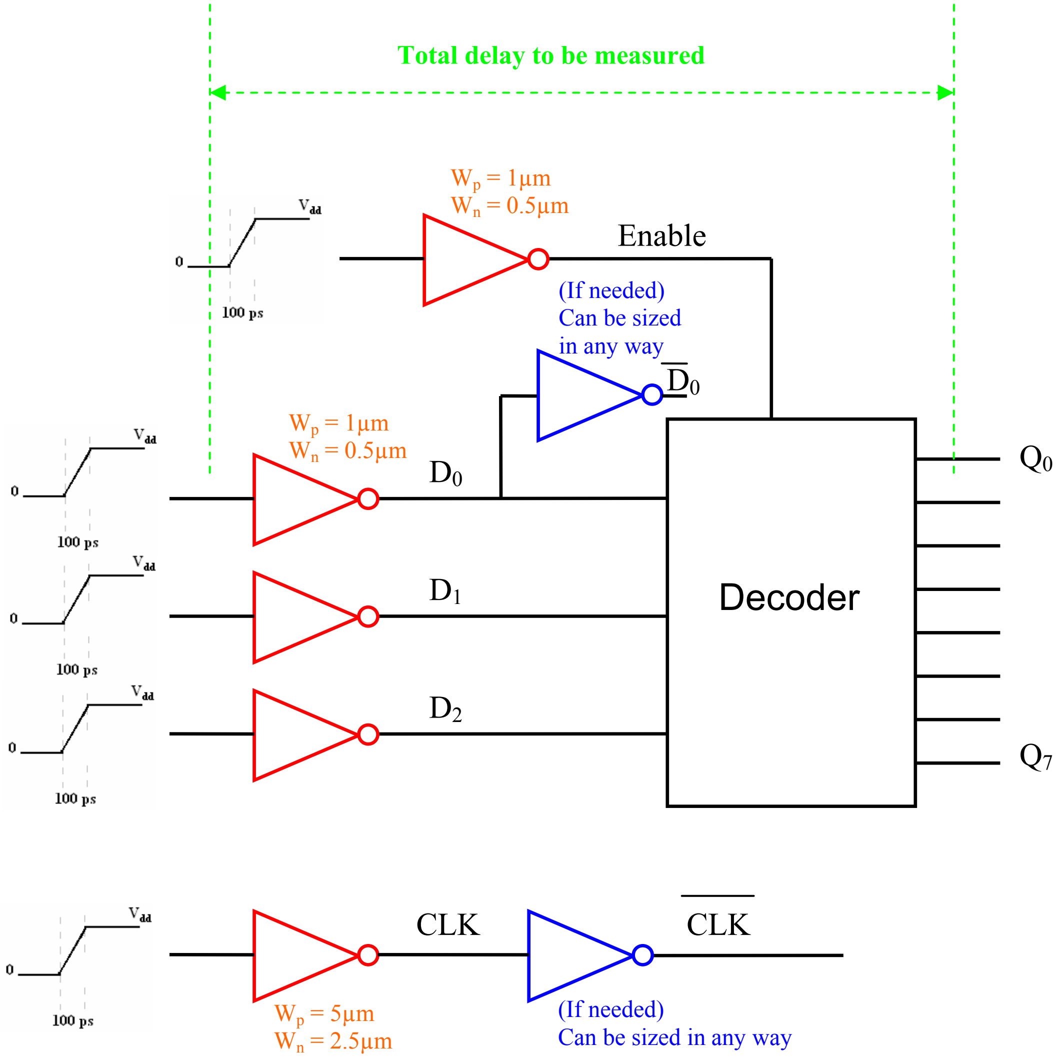

**Figure 1. A system perspective**

## Design Choices

### Supply Voltage

You **do not** have to use $V_{dd} = 2.5$ V in your design. You are free to choose any supply voltage ≤ 2.5 V (however you should use only 1 value in your design).

### Threshold Voltage

You have a choice of using two different threshold voltages for your design. The existing model file has been modified and it contains 4 device models. nhvth/phvth are NMOS/PMOS models with high threshold (these are the same NMOS1/PMOS1 devices you have used in your labs this semester). The low threshold devices are called nlvth/plvth. To select from among these devices, go to Libraries > MGC Digital Libraries > Display Library Palette in Design Architect and select EECS312_lib. This library contains some of the basic elements that you will be using often. The high $V_{th}$ devices have the default (blue) color and the low $V_{th}$ devices are orange. When you select these devices, the default AS, AD, PS, PD, W, and L parameters and model names have been predefined. All you need to modify in the properties dialog box is the width and length ( you do NOT need to compute AD, AS, etc. as these are computed for you whenever you alter the transistor width). To use elements other than transistors/vdd/gnd/portin/portout, (e.g., a capacitor) you can use the same library as you did for your labs. When selecting the model (high Vth or low Vth), you should keep in mind the implications of using a reduced $V_{th}$ transistor on speed and power.

### Logic Family

You are free to use any logic style such as static CMOS, pseudo-NMOS, domino, pass-transistor, or any combination of logic families. (The minimum allowable width for devices in this process is 0.4μm and minimum allowable channel length is 0.25μm).

### Input drivers

All inputs are driven by static CMOS inverters with Wp=1μm and Wn=0.5μm (you must measure the total circuit delay from the input of these inverters although technically this is the complement of the inputs, see Figure 1) and inputs to these inverters have a risetime and falltime of 100ps. These inverters are shown in red in Figure 1 and are required. (Suppose your design has inputs A, B, C …, then these would be the *output* of an inverter with the above size. If needed, the complement of the inputs must be generated by using another inverter whose size you can choose. These optional inverters are shown in blue in Figure 1).

The input driver for the clock signals have Wp=5μm and Wn=2.5μm (it must be able to drive the large capacitive load on clock signal). Again, if the complement of clock is needed, another inverter (size of your choosing) must be cascaded.

### Output Load

Each of the 8 output bits has a lumped capacitive load of 200fF.

## Design Evaluation

**Energy Measurement:**
You will need to measure the current drawn from $V_{dd}$ over all possible input patterns. Note that static power will count towards this so if your design results in significant static current to enhance speed, it will hurt you here. The input test pattern to measure average energy consumption will be provided to you in the form of an Accusim dofile. Be sure to name your inputs D0, D1, D2, E (enable), CLK (in case of dynamic logic). You may need to alter this dofile slightly depending on the exact implementation of your decoder (this will hopefully be relatively straightforward; details can be provided if needed).

**Clock**:
If you use dynamic logic, a precharge/evaluate clock can be defined (if you need two phases of the clock or more, you must generate these yourself – do not define a CLK and CLK_bar input directly in Accusim). Clock frequency is 400MHz with a 50% duty cycle (% of time the clock is high). The result must be available after one clock cycle (no pipelining; it's not needed in this project). Clock power is counted against the entire circuit power consumption. For a dynamic design, you can assume the following: the inputs are unknown during precharge phase (you do not know and cannot control their states during precharge; they may be either high or low) and when enable is low, clock does not toggle and is simply held high. The inputs can toggle any time after (or at the same time) as the clock edge when entering the evaluate phase to trigger an output transition. If you use a static design, no clock is necessary. Assume that the inputs to the clock input drivers transition at 100ps (0-100%).

**Design Robustness**:
Outputs must swing rail-to-rail. Noise margins must exceed 10% of $V_{dd}$. You can check this by entering $0.1V_{dd}$ and $0.9V_{dd}$ values at inputs; if the circuit still functions properly your noise margins are OK. Transition times (10-90%) at the outputs should not be larger than 400ps. Inputs are not available in complementary form.

**Wire capacitance**.
To consider the impact of wiring, you will need to add a lumped capacitance in Accusim at the output of each gate. The values to use are functions of the fan-out of the gate. For instance, if an inverter drives 2 NAND gates and 1 NOR gate, it has a total fan-out of 3. The wiring capacitance is assumed to be 7fF for a fan-out of 1. For a fan-out of N, let the wiring capacitance be given by: 7fF*(1+0.4*(N – 1)). Example: fan-out of 3 gives $C_{wiring} = 7*1.8 = 12.6$fF. These capacitances must be added to the clock as well if you use one.

## Design Documentation

**Report**:
The goal of the report is to describe your overall design approach and document your results. This includes describing why you chose the logic family you did, what logic simplifications you made, how you went about meeting the timing and reducing energy*delay product. Provide initial estimates (hand calculations of speed and energy) and show how they compare to the final

simulated results (it's OK if they don't match very well; some explanation could be useful in these situations though). Make sure you show transistor sizing on schematic diagrams. Clearly mark the critical path (longest delay path) of your circuit, including input patterns used to test for the worst-case delay. You need to justify the design choices ($V_{dd}$, $V_{th}$, etc.) and their implication on the performance. You must demonstrate that your design is completely functional for all possible input patterns. Show waveforms for all input combinations and resulting outputs. On the front cover of your report, list the worst-case delay (in ps), the energy*delay product (in pJ·ps), and the supply voltage used (in V). Finally you need to mention the contribution of each group member in the project.

Limit your report to 6 pages including plots (embed these in the text). Use hand calculations first, based on our default technology, to determine rough sizings, $V_{dd}$ values, gate topologies. Plots to include are transistor-level schematics, waveform diagrams showing functionality, and worst-case delay and average energy (based on current drawn from global $V_{dd}$). Also demonstrate that the circuit's noise margins are sufficient. You will need to shrink some of these plots to save space. Try to keep them somewhat legible. Be concise in your descriptions of your design decisions. This is an important attribute to learn – you should be able to convey all the essential things about your design in a reasonably small amount of space. This demonstrates you have a good grasp of what is important and not important.

The quality of the report is a very important factor in your final grade. Do not turn in a project report that looks like your software lab reports; these are too rough and unpolished. Take some time to prepare a professional looking report in a word processor such as MS-Word, Framemaker, etc. with embedded figures, a catchy title, and even references if you used any. Create a lab subdirectory in one of the group members class directories. Include the path to this directory in your report.

**Grading criterion:**
Results (these will be measured in part by comparing across the class): **30%**
Correctness and sound design approach: **30%**
Creativity: **15%**
Report quality: **25%**

---

## Details on creating symbols and exploiting hierarchy in Design Architect

Large schematics can be set up by using different building blocks, such as inverters, NAND gates, NOR gates etc. They can be created as symbols using *Design Architect.* To create a symbol for use in other Design Architect schematics, follow these steps (described for an inverter):
After the schematic capture for inverter is done and saved, we need to make a symbol for the inverter. Select the following.
***Miscellaneous > Generate Symbol...***

Choose ***Yes*** under the *Replace Existing* area in the lower left corner and then click on the *OK* button in the dialog box. A symbol for the design appears. This contains the pins *in* and *out* and their types i.e. in for input and out for output pins. Once again the symbol needs to be checked. Select:

***Check > With Defaults***

The first time you check a new design you get a warning saying that pins (*in* and *out* in this case) are not present on the interface. Ignore it, since it will go away after you have saved the symbol. Close the report window and save the symbol by selecting:

***File > Save symbol > Default Registration***

The above completes the creation of the schematic and its corresponding symbol. In future designs you can change the symbol shapes to your liking. If you do this, try not to delete any of the pins or the pin properties while editing the symbol. If you do and you're unable to recreate them correctly, delete the symbol using Design Manager and regenerate the symbol.

As you are building higher level schematics, you can select ***Choose Symbol*** in the right menu. Then select the right symbol in the list that pops up. You can wire together symbols in this way so that complex schematics are clarified.