EEE3096S: Embedded Systems II

LECTURE 22:

TOWARDS HARDWARE DESCRIPTION LANGUAGES

Presented by:

STANLEY MBEWE

Electrical Engineering
University of Cape Town

OVERVIEW

- Recap of Digital Logics … towards a 1-bit CPU
  - The Mux and Adder (e.g. of ALU element)
  - Precursor to CPU architecture
- Introduction to FPGAs and HDL

PLANNING NOTE: main thing is to look over **Appendix A** of textbook as a reminder of digital logic aspects that you've probably encountered, at least substantial parts of that content, in prior courses such as EEE2046F.

Connection to Text Book*
The preliminary topics connect with **CH3 3.1 and 3.2 pp. 188 – 191**, you need not go into the multiplication details, that is picked up in EEE4120F if you want to do that course that covers digital logic and further HDL. Also have first look over CH4 4.1, as planning to touch on datapath in last lectures of this course – although that is not needed for learning FPGA and HDL related topics covered this and next week.
NB: You're expected to know most of the digital logic aspects covered in **Appendix A** of the text book.

*Recommended textbook for this course is "Computer Organization and Design - ARM Edition" by A. Patterson, John L. Hennessy

Context Switch to: thinking about Digital Logic

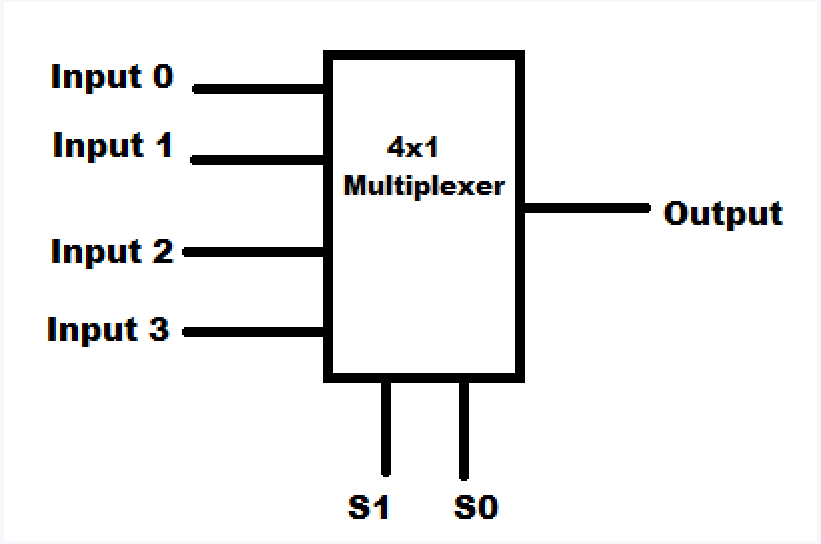
A look at two essential aspects (besides the very basics) that are used in Digital Logic circuits …

Acknowledgement: digital logic slides adapted from J. Wyngaard’s 2020 EEE3096S lectures

WHAT IS A MULTIPLEXER?

- Defn: Multiplexer:

  This is a combinational logic circuit that switches one of several input lines to a single common output line according to state of its control signal.



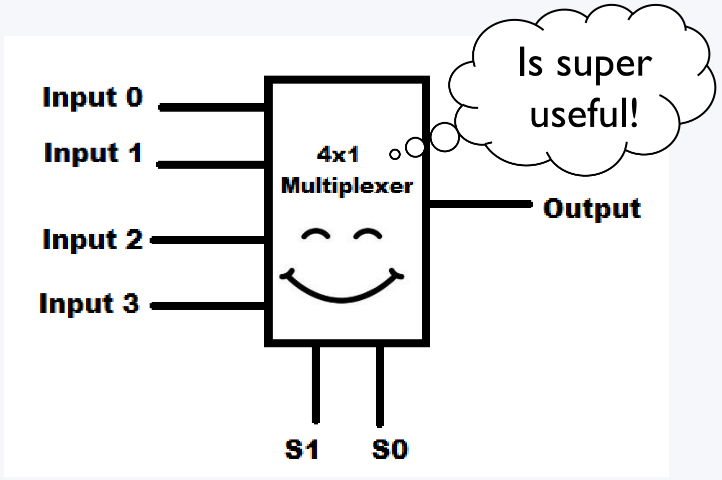
Consider input 0 to 3 are digital input lines, one of which you want to connect over the Output line, and leave the other inputs on high impedance (usually done in FPGAs or digital designs) or unconnected. It would technically be a digital multiplexer if you are only dealing with digital (usually binary) signals.

WHY EMPHASISE MULTIPLEXER?

- Why recap Multiplexer first?

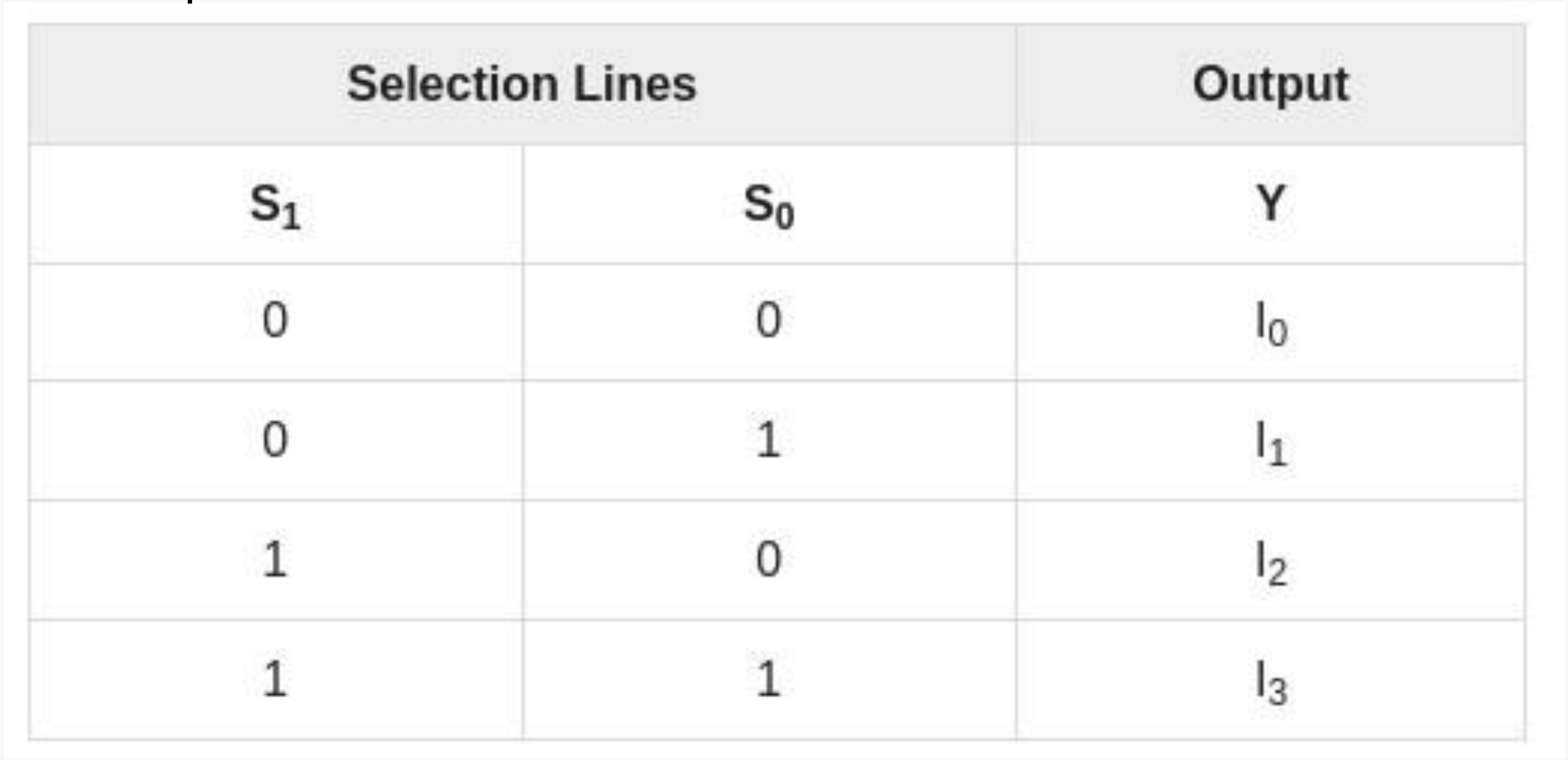
  Because it’s a super useful circuit! And is essential in many digital designs, e.g.:

  - in ALU: for choosing a function, what inputs to use and where to put result
  - in State-Machine: apply action for current state, decide next state depending on input, next state



TRUTH TABLE (TEMPLATE) OF MUX

Not that you really need this slide, as it's very easy to reconstruct what the output would look like. The output needs to be set to what the value of the selected input line is.

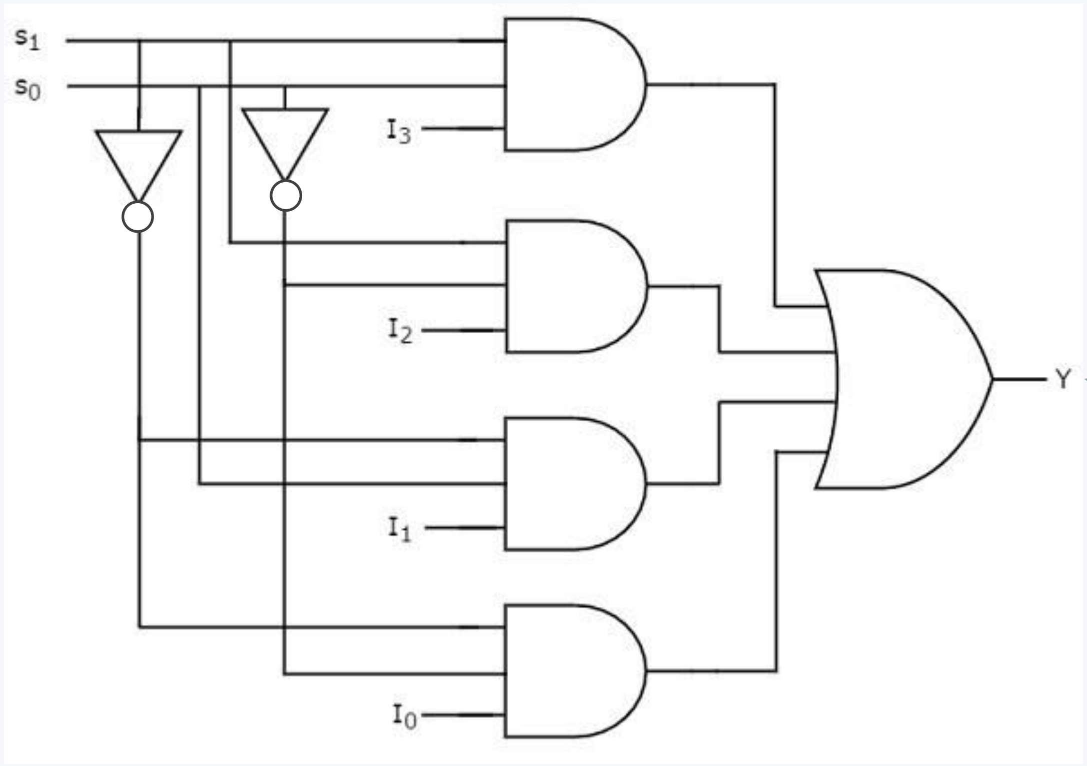


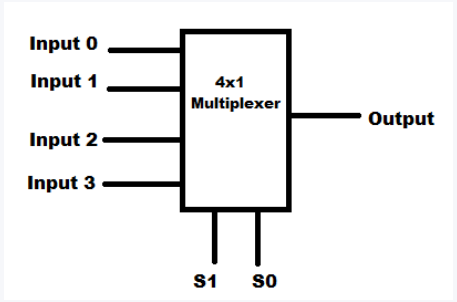
Selection Lines		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

BUT what is potentially very useful is figuring out what the digital circuit (using basic gates, as in AND, OR, NOT) would look like. (i.e. a hint for tests 😉)

SIMPLE 2-SELECT LINE MUX

S: Select input





MUX component symbol

Y: output bit

I_0…I_3: inputs

Underlying gate design of a 4x1 digital MUX

Only one of the inputs are connected through to the output

You're expected to provide sufficient inputs, otherwise just tie the line to 0 or 1 (alternatively to high impedance to avoid potential of shorting, although a design tools would not allow 0 / 0V connected to 1 / +VCC anyway).

ADDER CIRCUIT

How do you do a digital circuit that does: X + Y ?
i.e. how do you implement an **adder**? …

But why?...

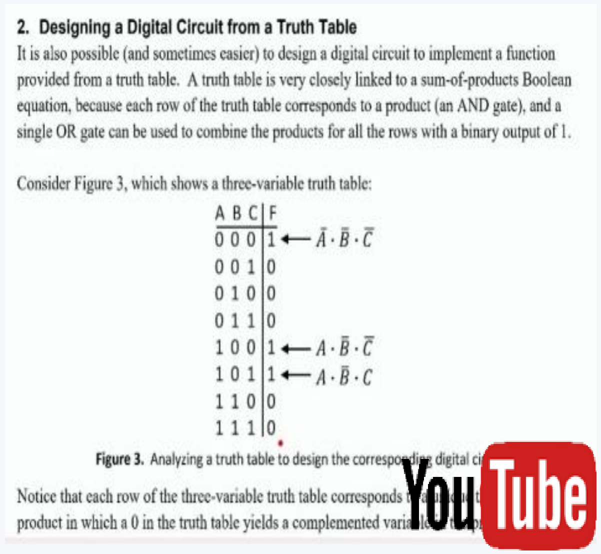
Consider it as an example of what component is used for processing in a compiler system. The scenario of a MUX selecting operations further…

DESIGNING AN ADDER

- The usual process for designing a (simple) gate-based combination logic circuit is to:
  - Develop the truth table for the circuit
  - Derive a Boolean expressions from the truth table (i.e. for cases where output=1)
  - Simplify the Boolean expression
  - Draw up the circuit

If you need a refresher, please look over this YouTube clip:
https://youtu.be/sS1FJ_Kab9w

(hence this is aimed as a refresher for you.)



2. Designing a Digital Circuit from a Truth Table

It is also possible (and sometimes easier) to design a digital circuit to implement a function provided from a truth table. A truth table is very closely linked to a sum-of-products Boolean equation, because each row of the truth table corresponds to a product (an AND gate), and a single OR gate can be used to combine the products for all the rows with a binary output of 1.

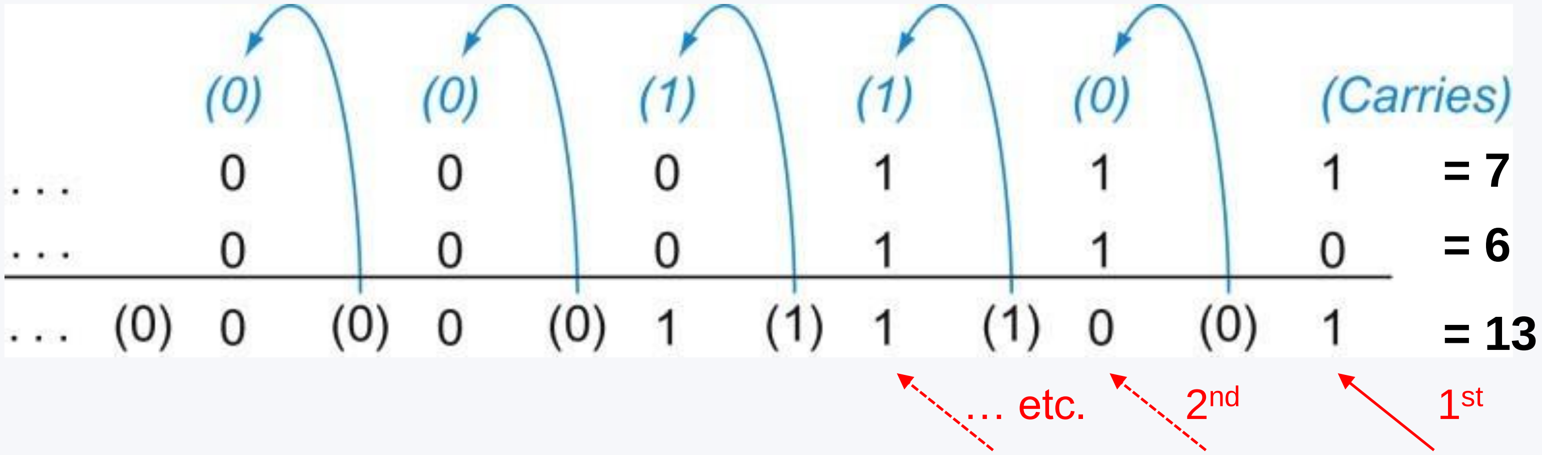
Consider Figure 3, which shows a three-variable truth table:

A	B	C	F	
0	0	0	1	$\leftarrow \bar{A} \cdot \bar{B} \cdot \bar{C}$
0	0	1	0	
0	1	0	0	
0	1	1	0	
1	0	0	1	$\leftarrow A \cdot \bar{B} \cdot \bar{C}$
1	0	1	1	$\leftarrow A \cdot \bar{B} \cdot C$
1	1	0	0	
1	1	1	0	

Figure 3. Analyzing a truth table to design the correspo[illegible] digital ci

Notice that each row of the three-variable truth table corresponds [illegible] product in which a 0 in the truth table yields a complemented varia[illegible]

ADDER OPERATION



e.g.: 7+6 = 13

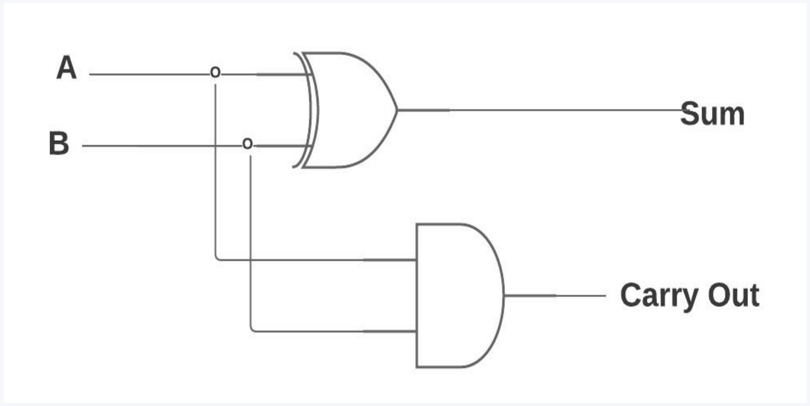
Process starts at LSB, but *note*: if a Carry In need to do that first.

ADDER DESIGN

- A common, and very modular, approach is to build a full adder from two half adders

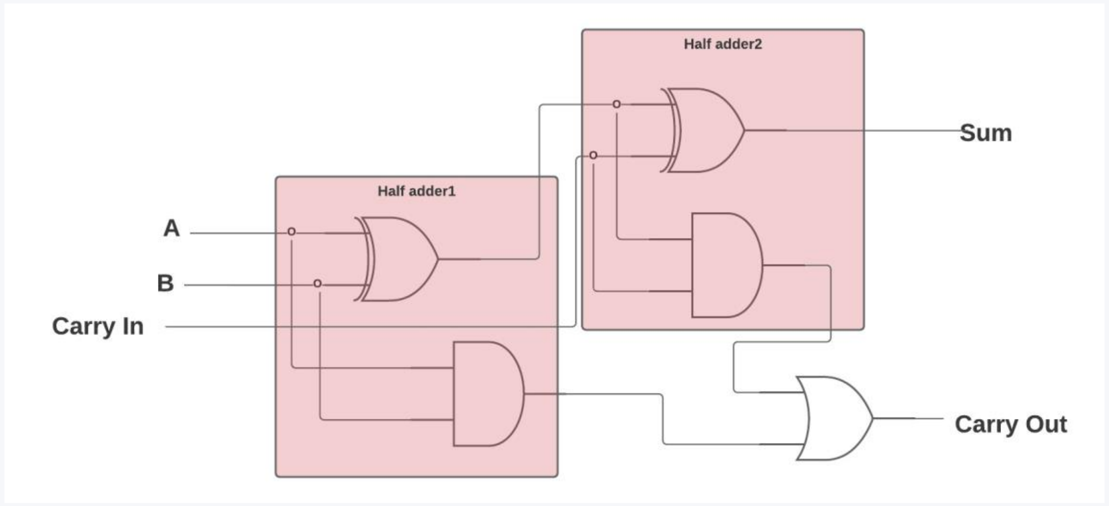
Truth Table for Half-Adder

Inputs		Outputs	
A	B	Carry out	Sum
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

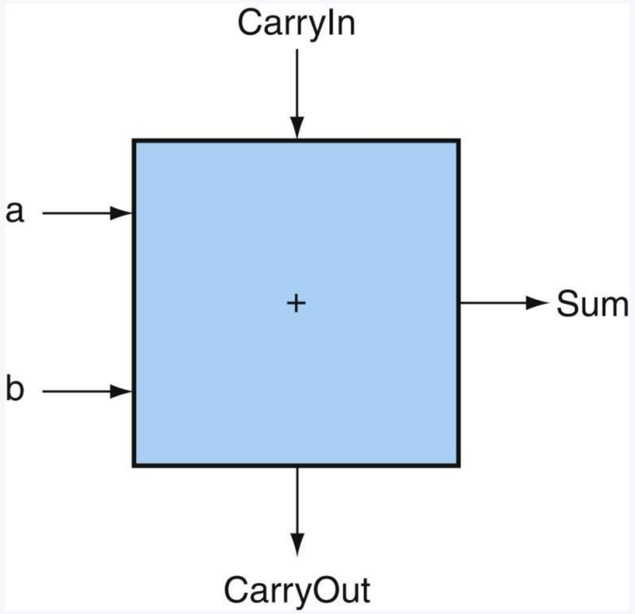


FULL-ADDER DESIGN

Inputs			Outputs	
A	B	Carry in	Carry out	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



FULL-ADDER CIRCUIT ELEMENT



A 1-bit adder. This adder is called a full adder; it is also called a (3,2) adder because it has 3 inputs and 2 outputs. An adder with only the a and b inputs is called a (2,2) adder or half-adder.

Now Fairly Ready for HDL & FPGAs!