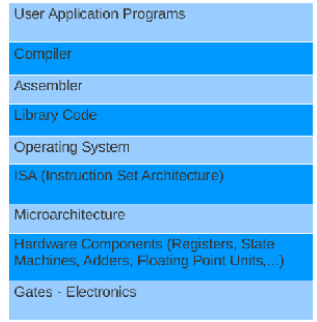
Digital Building Blocks

Eric McCreath

A Layered Approach

- A computer system can be divided up into layers.
- Each layer is build upon the resources the layer below provides.
- This is a common design approach for complex systems, as it simplifies and partitions the overall design.



Digital Logic

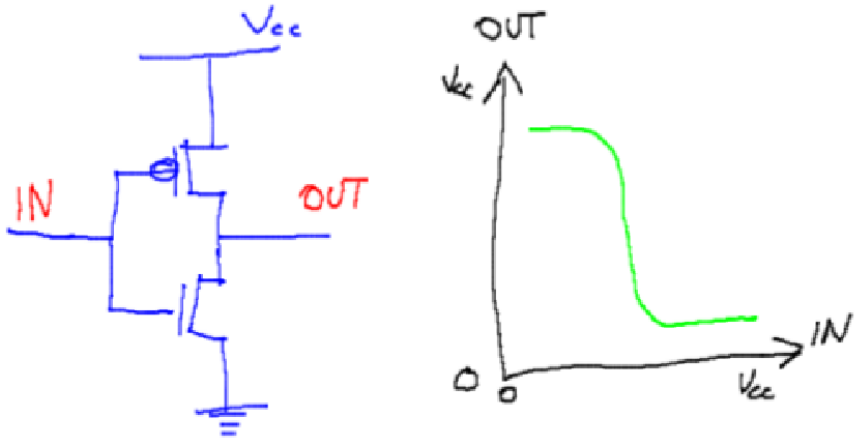
- Digital logic is concerned with the design of components such as: memory, adders, instruction decoders,.... Hence, it is foundational for computer hardware.
- The electronics provide the basic gates. These gates are combined together to form the components.
- The logic is simple and based on Boolean logic. George Boole (1815-64) was first to have framed this logic as an algebra.

The Transistor

- The invention of the transistor in 1947 opened up the way for devices with millions of switching elements.
- There are a number of different commonly used technologies. These include:
  - Transistor-transistor logic (TTL) family, and
  - Complementary Metal Oxide Semiconductors (CMOS).
- These different technologies all provide basically the same logic gates.

The Transistor Switch

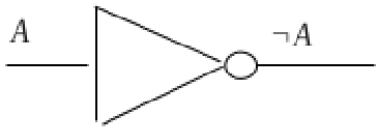
- The transistor can act as a switch. The switch is a key component in the construction of logic gates.



A not gate.

not

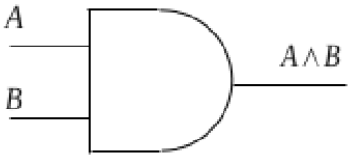
- The 'not' operator evaluates to true when the input is false, otherwise it evaluates to false.



A	¬A
0	1
1	0

and

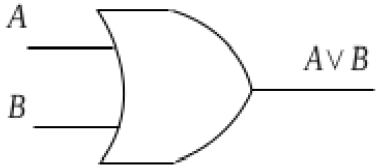
- The "and" operator evaluates to true exactly when all its inputs are true.



A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

or

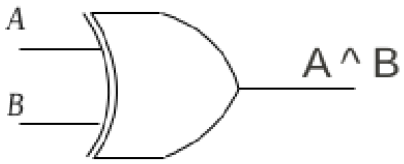
- The 'or' operator evaluates to true when any of its inputs are true. (false otherwise)



A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

xor

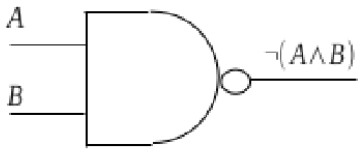
- The 'xor' operator evaluates to true when exactly one of its inputs are true.



A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

nand

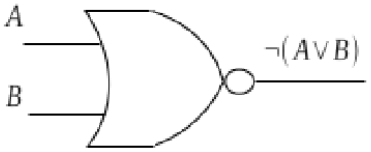
- The 'nand' operator evaluates to true when any of its inputs are false.



A	B	$\neg(A \wedge B)$
0	0	1
0	1	1
1	0	1
1	1	0

nor

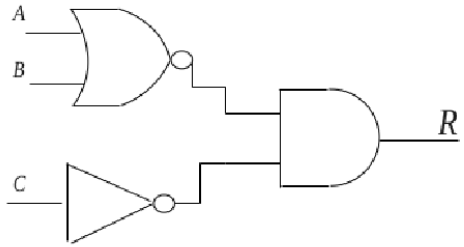
- The 'nor' operator evaluates to true exactly when all of its inputs are false.



A	B	$\neg(A \vee B)$
0	0	1
0	1	0
1	0	0
1	1	0

Design to Truth Table

- By using the truth tables of the basic gates we can always construct a truth table from a circuit design.



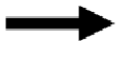
A	B	C	R
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Truth Table to Design

From any truth table we can also design set of gates that will evaluate the truth table.

Note that, this design process goes beyond the scope of this course.

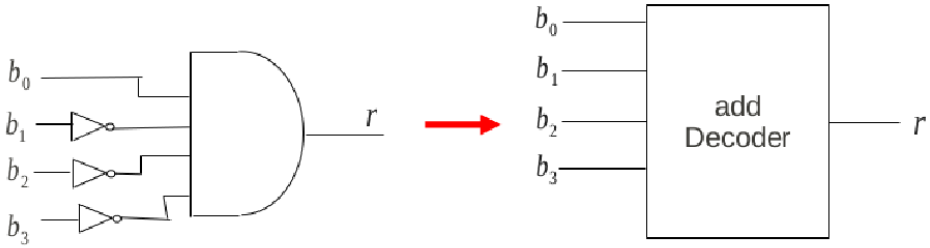
A	B	C	R
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



?

Instruction Decoder

Instructions are given a particular encoding. The architecture must be able to determine if a particular instruction has been loaded. An instruction decoder can do this. Suppose the instruction "add" has the encoding 0001 then the following circuit could decode it.



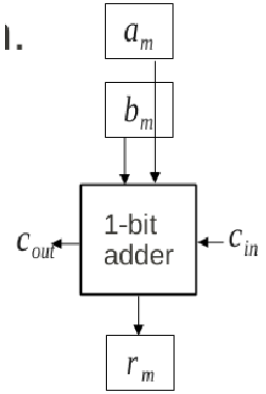
Binary Adder

- Binary addition is an important operation required by most architectures.
- When we add binary numbers by hand we place one number on top of the other. Then we move from right to left adding the digits and remembering to include an carry. This exact approach can be replicated in digital logic.

```
  00110
+ 00101
--------
  01011
```


Binary Addition

- We first create a component that will evaluate one column of the addition.
- The inputs will be a single bit from each binary number of the column in question. Also the carry in will be provided to this component.

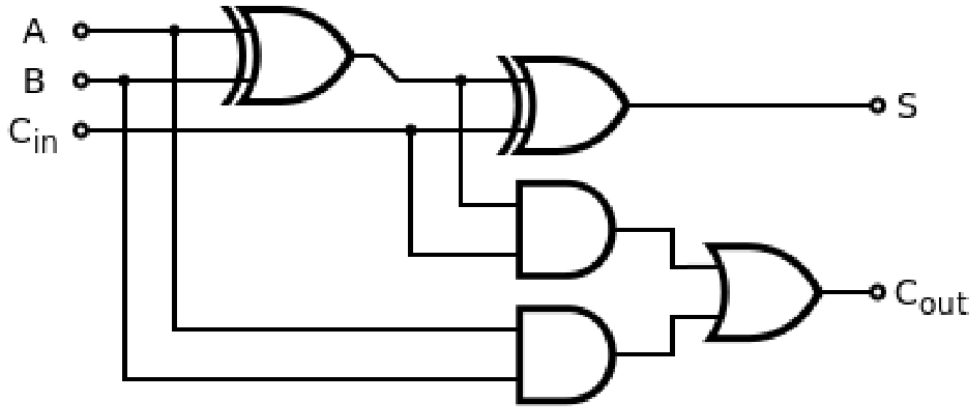


Binary Addition

- We can construct a truth table for this one bit adder. From this truth table we can then design a circuit.

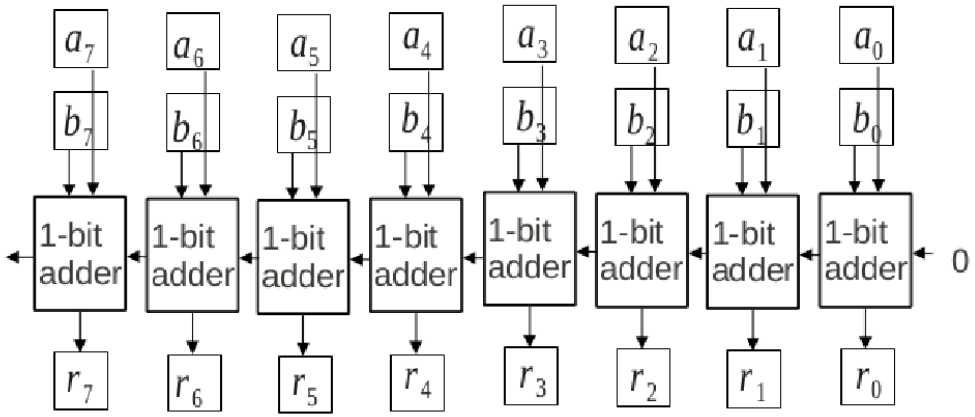
a_m	b_m	c_{in}	r_m	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adder



8 bit adder

We can combine 8 of these 1-bit adders to form a serial 8-bit adder.



Exercises

Work out the circuit for the below truth table:

A	B	C	Out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Work out the truth table for the below circuit: