Slide 1

Systems and Devices 1
Lec 3a :
Combinatorial Logic

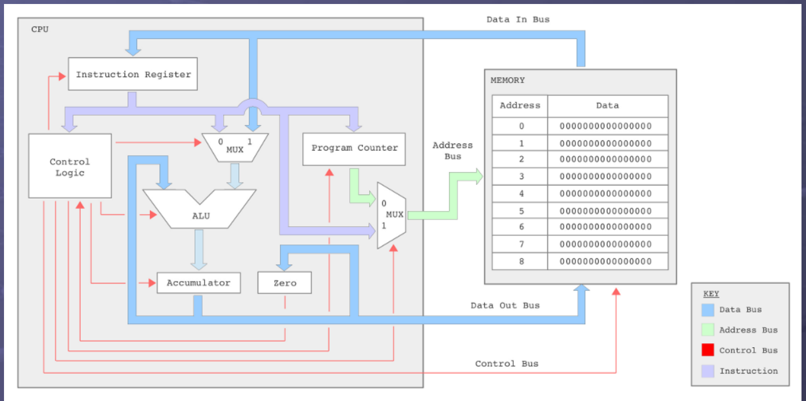
Slide 2

Before we get started ...

- We live in the fourth generation of computers:
  - Thermionic value, Transistor, IC, Micro-processor
- Default technology: transistor, therefore preferred number base: binary, processed using Boolean logic gate.
- How do we process data within our computer?
  - Identify the state of the computer e.g. what operation it is currently performing, what data has been selected to be processed and what results are produced?

Slide 3

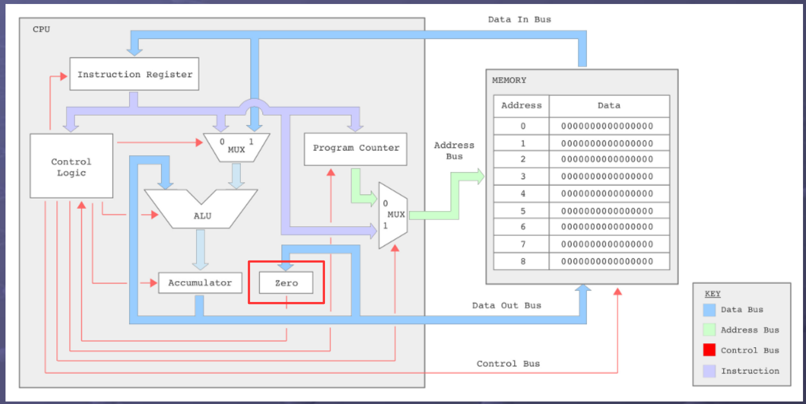
SimpleCPU_v1a



- Block diagram
  - We need to implement each functional block using logic gates i.e. as combinatorial logic circuits.

Slide 4

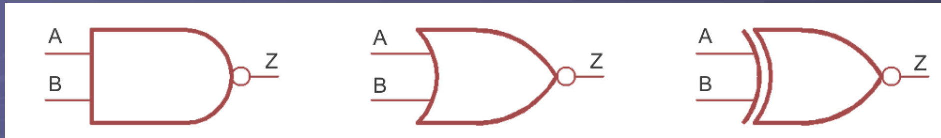
SimpleCPU_v1a



- Block diagram
  - Status bits : to control the flow of information in a computer we need to test its state i.e. data values.

Slide 5

Logic gates



NAND	A	B	Z
	0	0	1
	0	1	1
	1	0	1
	1	1	0

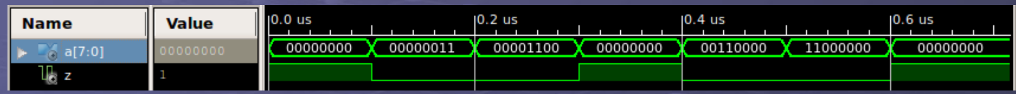
NOR	A	B	Z
	0	0	1
	0	1	0
	1	0	0
	1	1	0

XNOR	A	B	Z
	0	0	1
	0	1	0
	1	0	0
	1	1	1

- Quick quizzz: what logic gate could be used to test if :
  - Two bits are equal A = B
  - Two bits are both zero A = B = 0
  - At least one bit is zero A = 0 OR B = 0

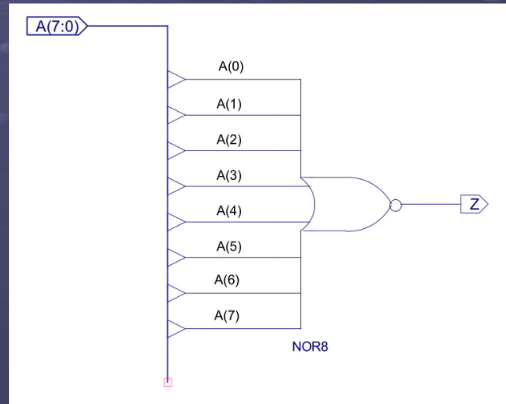
Slide 6

Example : NOR_8.zip



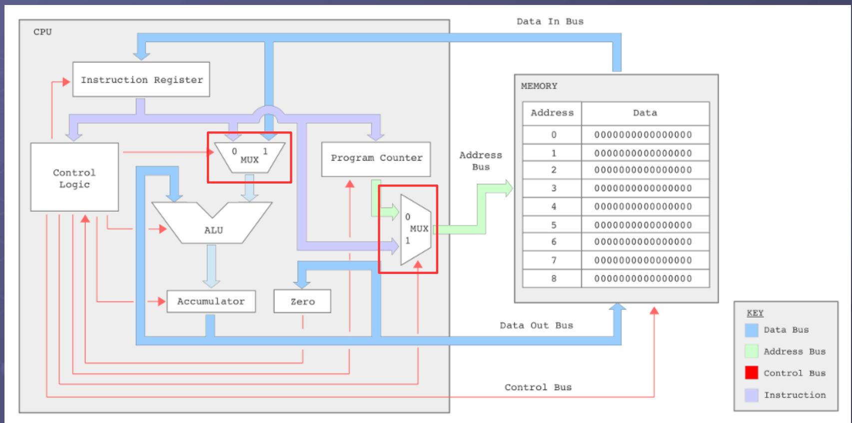
- To test if an 8 bit value is zero we can use an 8 bit NOR gate.

NOR	A	B	Z
	0	0	1
	0	1	0
	1	0	0
	1	1	0



Slide 7

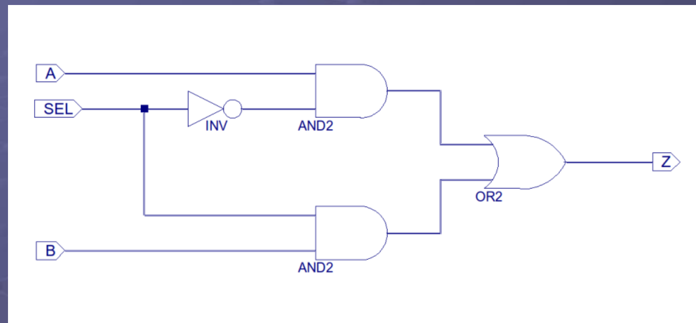
SimpleCPU_v1a

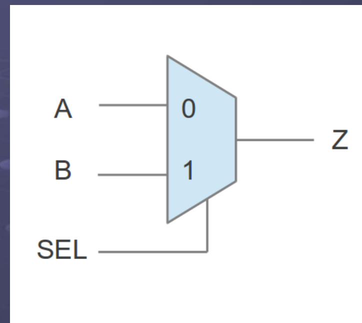


- Block diagram
  - Multiplexer : a core requirement of any computer is to route / move data between functional blocks.

Slide 8

Multiplexers

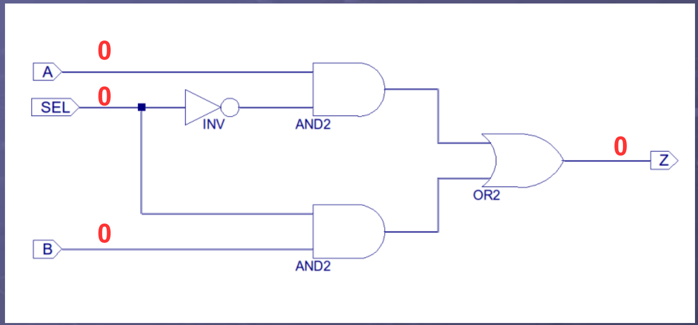


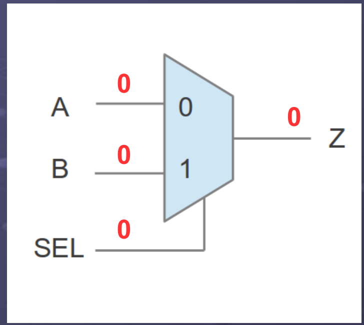


- 2:1 1bit data multiplexer (MUX)
  - Gate level implementation and Circuit Symbol
  - IF SEL = 0 THEN Z = A
  - IF SEL = 1 THEN Z = B

Slide 9

Multiplexers

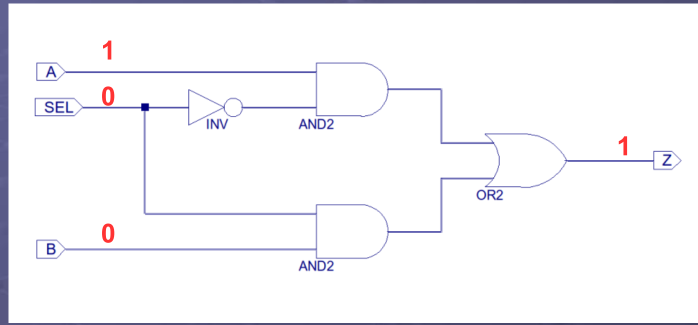


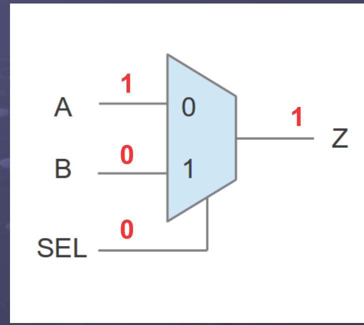


- 2:1 1bit data multiplexer (MUX)
  - Gate level implementation and Circuit Symbol
  - IF SEL = 0 THEN Z = A
  - IF SEL = 1 THEN Z = B

Slide 10

Multiplexers

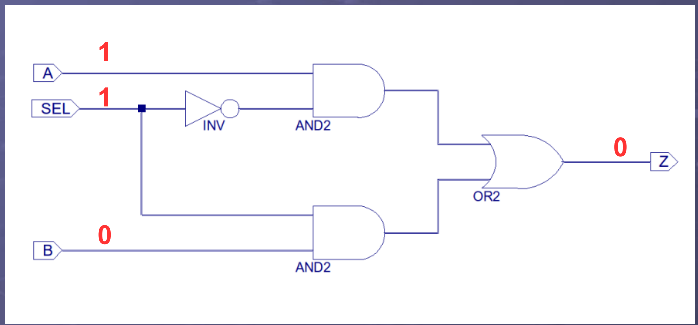


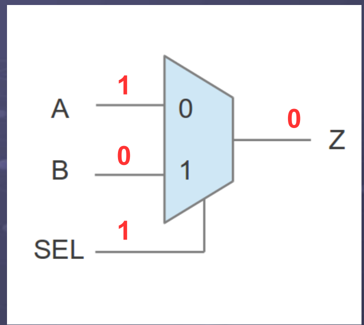


- 2:1 1bit data multiplexer (MUX)
  - Gate level implementation and Circuit Symbol
  - IF SEL = 0 THEN Z = A
  - IF SEL = 1 THEN Z = B

Slide 11

Multiplexers

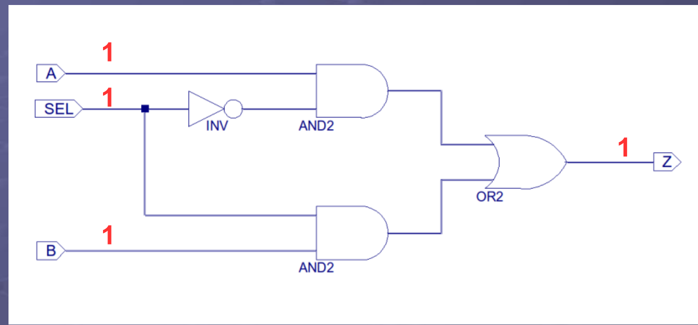


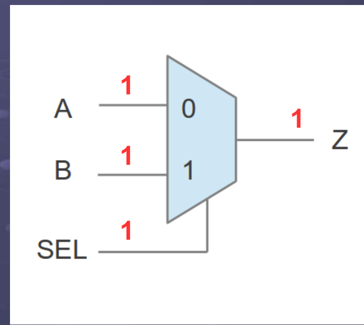


- 2:1 1bit data multiplexer (MUX)
  - Gate level implementation and Circuit Symbol
  - IF SEL = 0 THEN Z = A
  - IF SEL = 1 THEN Z = B

Slide 12

Multiplexers

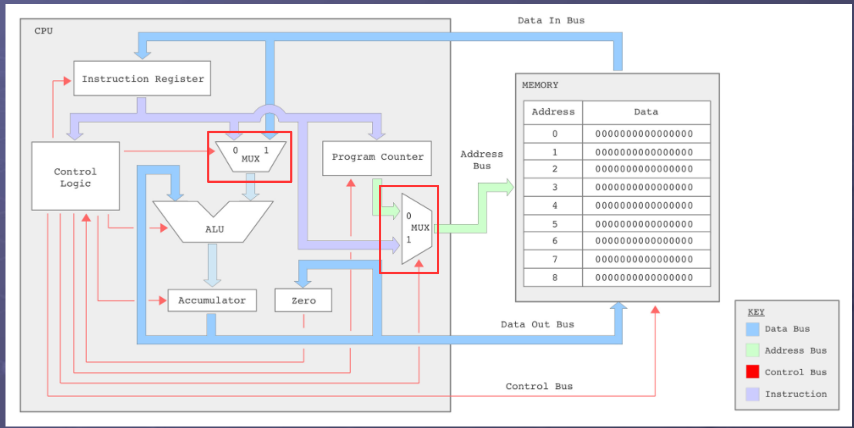




- 2:1 1bit data multiplexer (MUX)
  - Gate level implementation and Circuit Symbol
  - IF SEL = 0 THEN Z = A
  - IF SEL = 1 THEN Z = B

Slide 13

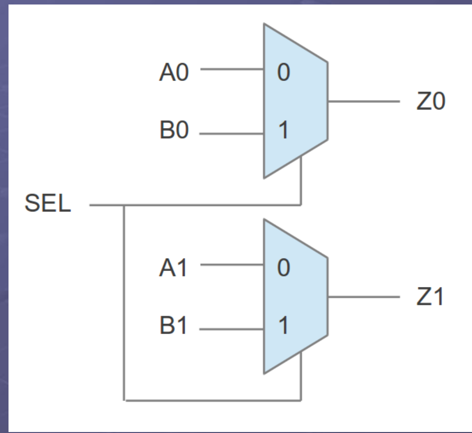
SimpleCPU_v1a

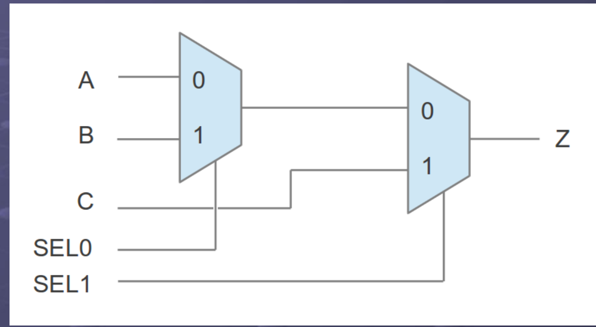


- Problem : the two highlighted multiplexers need to select between two 8 bit values.
  - May also need to select between more than two input sources.

Slide 14

Multiplexers

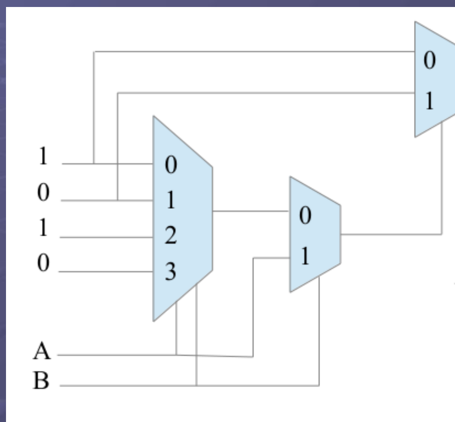




- 2:1 x 2bit MUX: Parallel MUX to increase width
- 3:1 x 1bit MUX: Serial MUX to increase inputs

Slide 15

Multiplexers

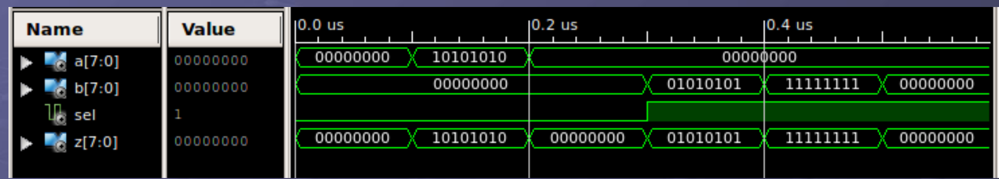


A	B	Z
0	0	?
0	1	?
1	0	?
1	1	?

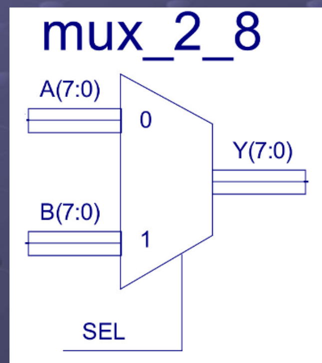
- Quick quiz: complete the truth table
  - Bonus question : what two logic gates can be used to implement this circuit?

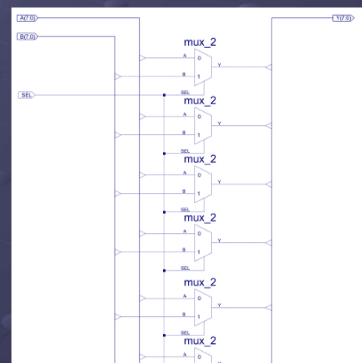
Slide 16

Example : MUX_2.zip



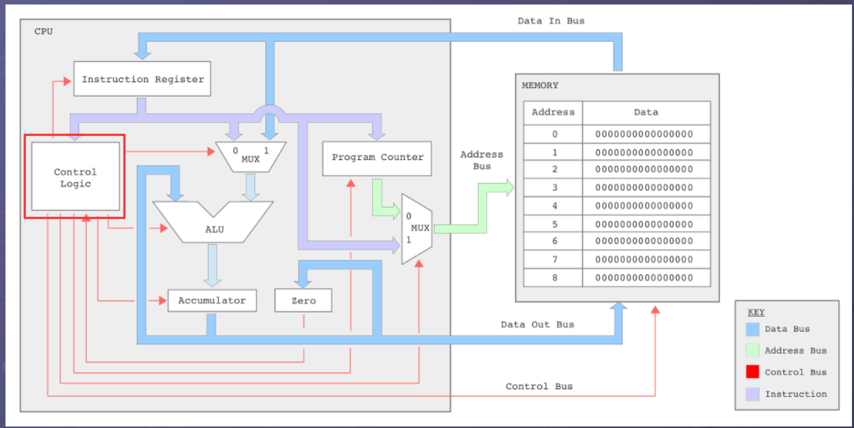
- Simulation
  - MUX_2_8 : a two input 8 bit multiplexer
  - MUX_4_8 : a four input 8 bit multiplexer





Slide 17

SimpleCPU_v1a



- Block diagram
  - Encoders / Decoders : within the processor information is stored using a range of binary representations.

Slide 18

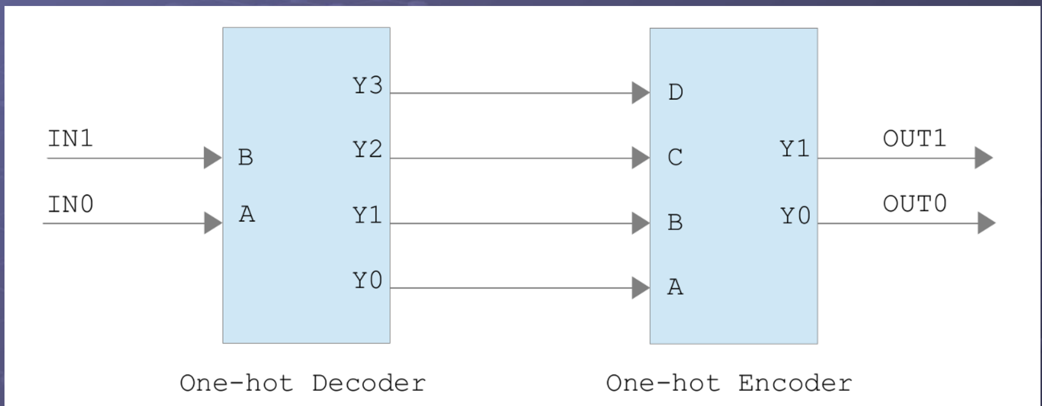
Binary encoding

Decimal	*Binary*	*BCD*	*One-hot*
0	0000 0000	0000 0000	0000 0000 0000 0001
1	0000 0001	0000 0001	0000 0000 0000 0010
2	0000 0010	0000 0010	0000 0000 0000 0100
3	0000 0011	0000 0011	0000 0000 0000 1000
4	0000 0100	0000 0100	0000 0000 0001 0000
5	0000 0101	0000 0101	0000 0000 0010 0000
6	0000 0110	0000 0110	0000 0000 0100 0000
7	0000 0111	0000 0111	0000 0000 1000 0000
8	0000 1000	0000 1000	0000 0001 0000 0000
9	0000 1001	0000 1001	0000 0010 0000 0000
10	0000 1010	0001 0000	0000 0100 0000 0000
11	0000 1011	0001 0001	0000 1000 0000 0000
12	0000 1100	0001 0010	0001 0000 0000 0000
13	0000 1101	0001 0011	0010 0000 0000 0000
14	0000 1110	0001 0100	0100 0000 0000 0000
15	0000 1111	0001 0101	1000 0000 0000 0000

- Alternative binary representations : Binary Coded Decimal (BCD) and One-hot encoding.

Slide 19

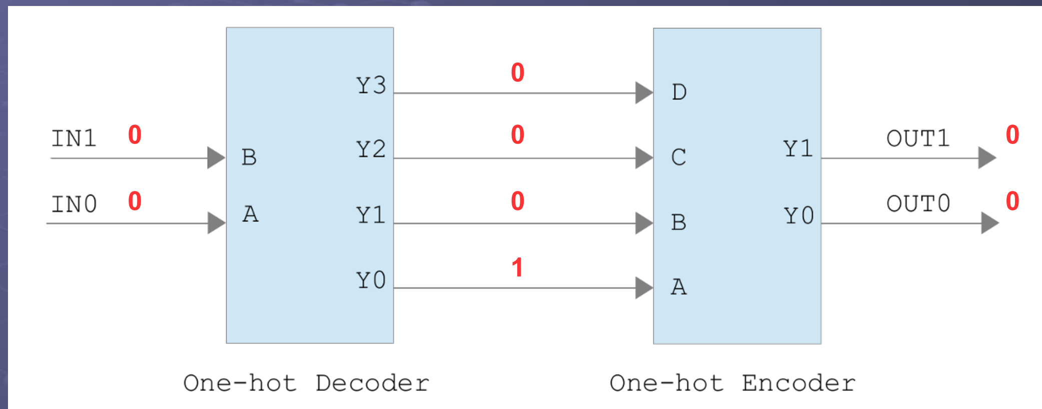
Encoder / Decoder



- A two bit binary to One-hot encoder and decoder

Slide 20

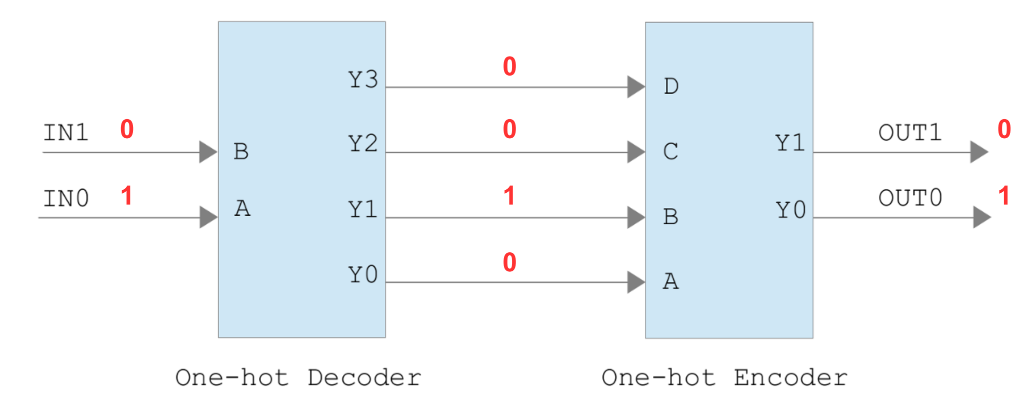
Encoder / Decoder



- A two bit binary to One-hot encoder and decoder

Slide 21

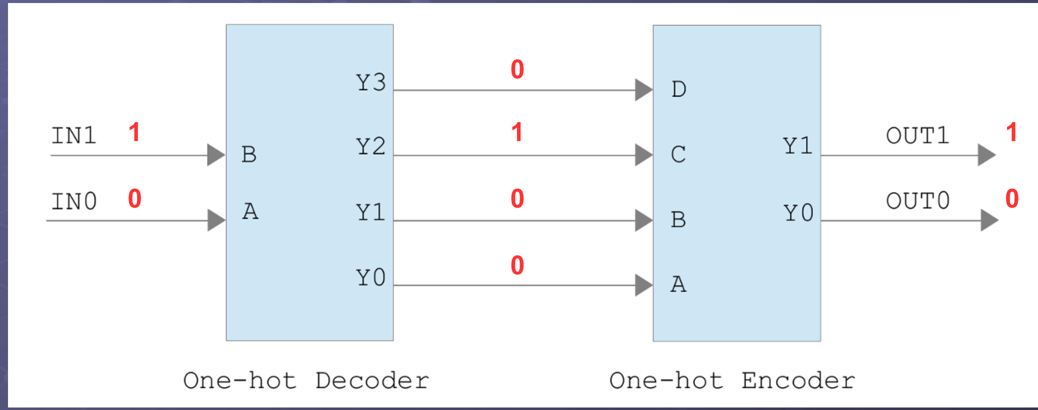
Encoder / Decoder



- A two bit binary to One-hot encoder and decoder

Slide 22

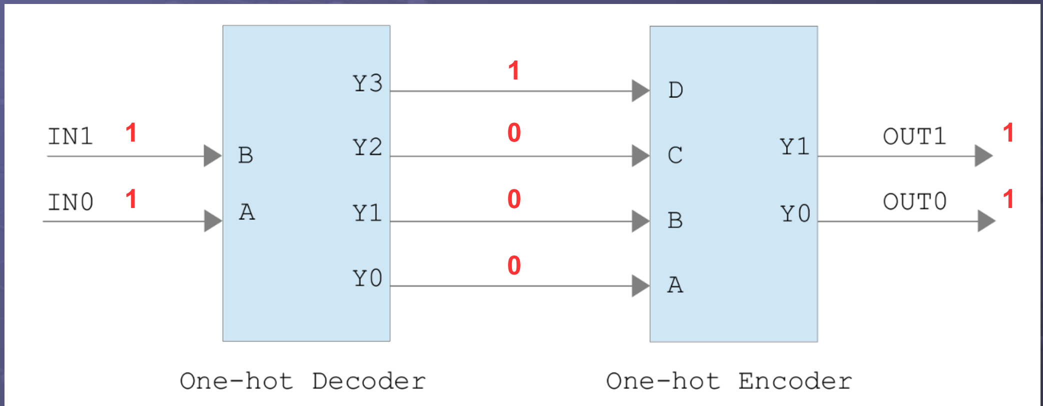
Encoder / Decoder



- A two bit binary to One-hot encoder and decoder

Slide 23

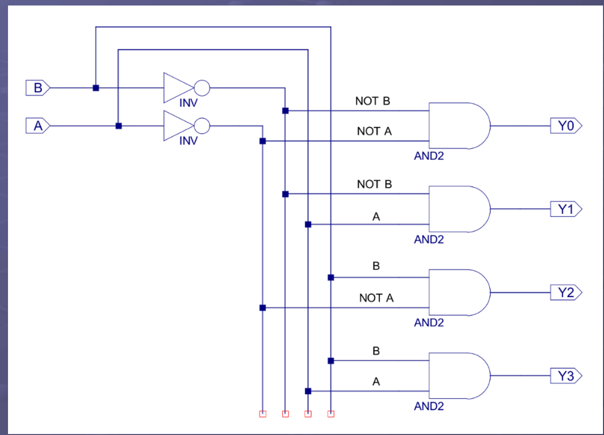
Encoder / Decoder

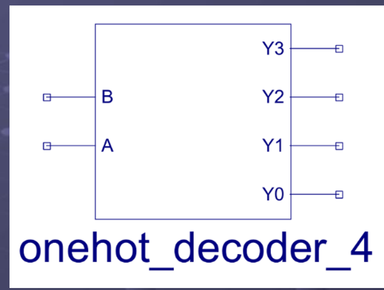


- A two bit binary to One-hot encoder and decoder

Slide 24

Decoder



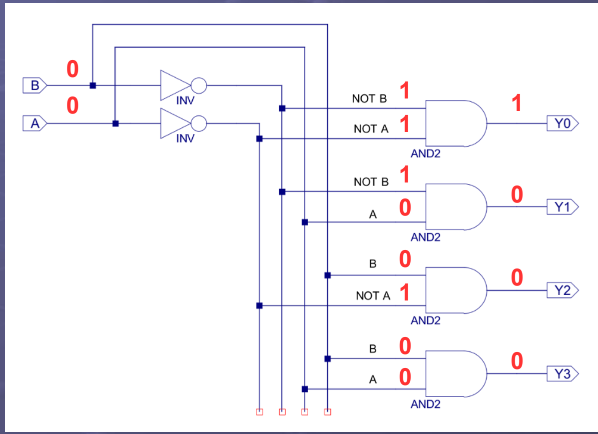


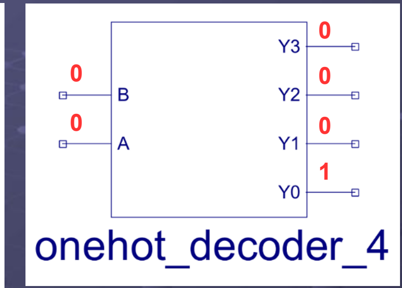
B	A	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- 2bit Binary to One-hot decoder

Slide 25

Decoder



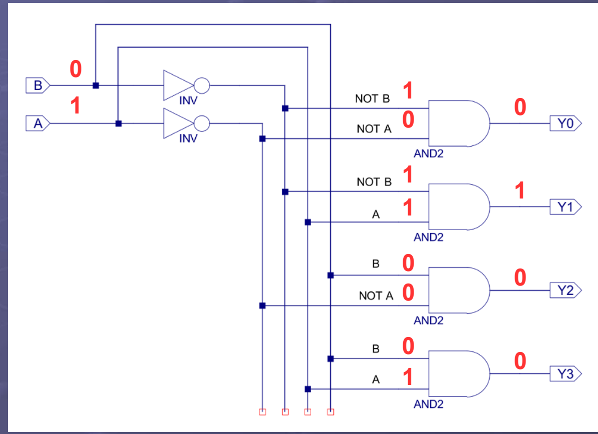


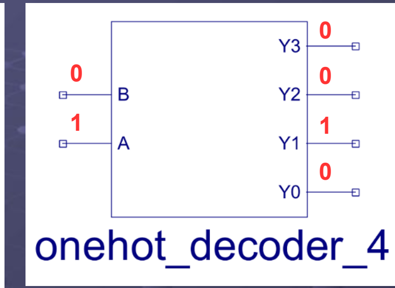
B	A	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- 2bit Binary to One-hot decoder

Slide 26

Decoder



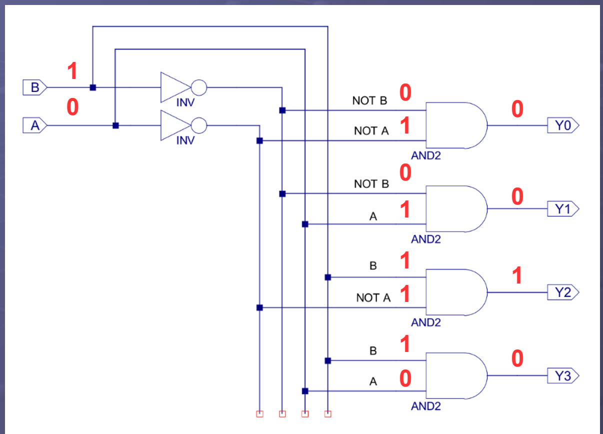


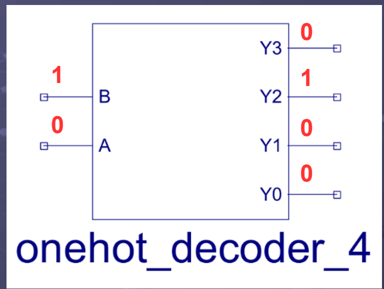
B	A	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- 2bit Binary to One-hot decoder

Slide 27

Decoder



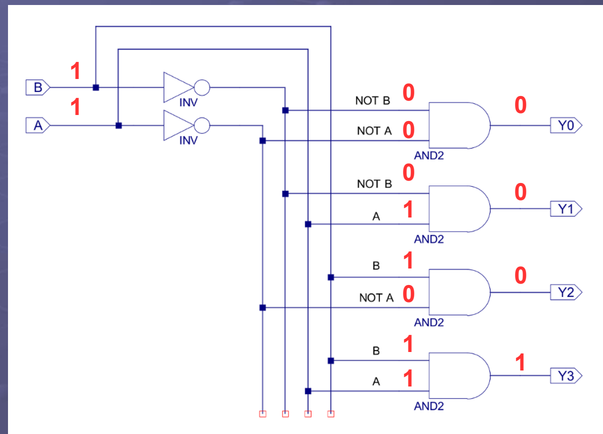


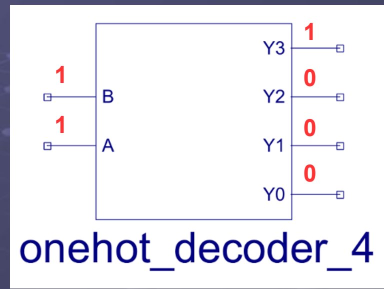
B	A	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- 2bit Binary to One-hot decoder

Slide 28

Decoder

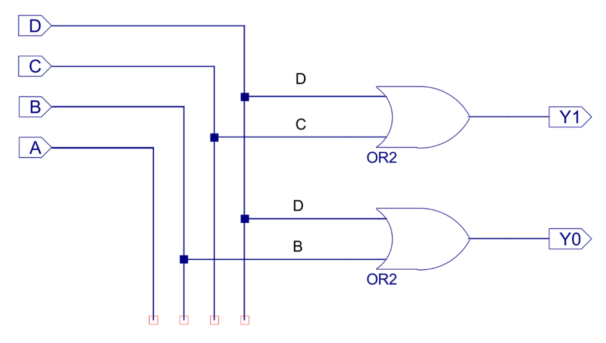


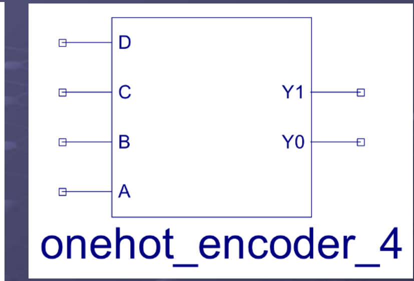


B	A	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- 2bit Binary to One-hot decoder

Encoder

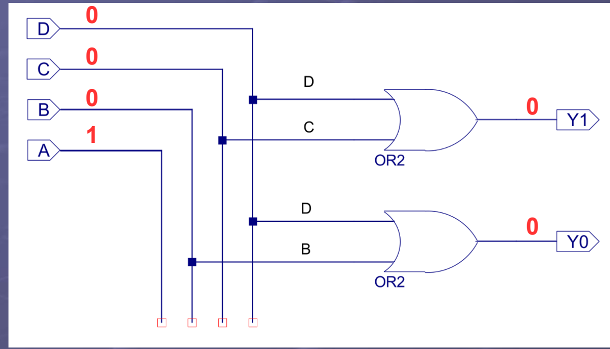


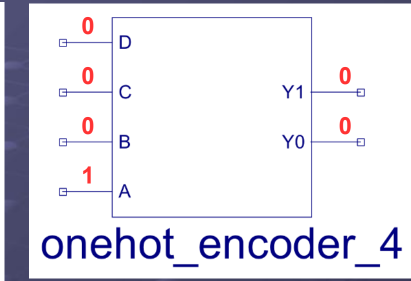


D	C	B	A	Y1	Y0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

- 2bit One-hot to Binary encoder
  - One-hot representation simplifies logic design i.e. only 1 bit is ever set.
    - Identify when output is a logic 1 then join active inputs with OR gates.

Encoder

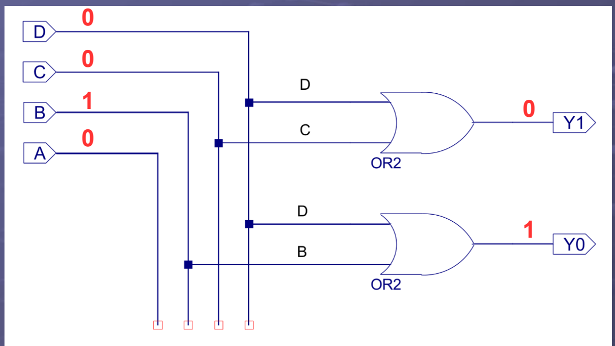


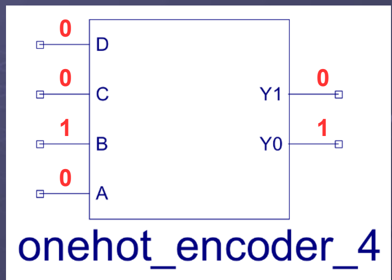


D	C	B	A	Y1	Y0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

- 2bit One-hot to Binary encoder
  - One-hot representation simplifies logic design i.e. only 1 bit is ever set.
    - Identify when output is a logic 1 then join active inputs with OR gates.

Encoder

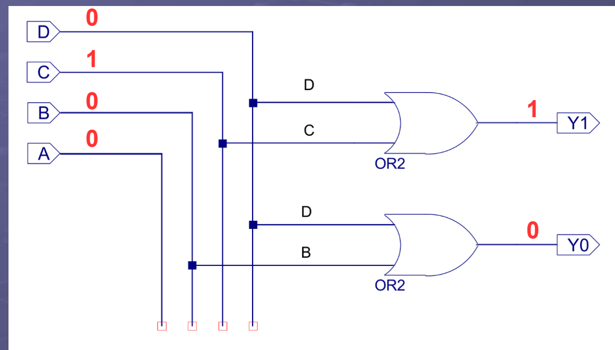


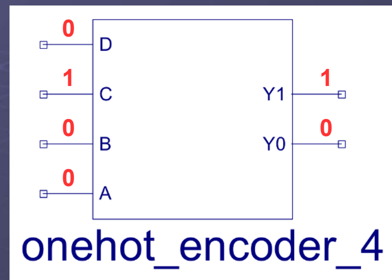


D	C	B	A	Y1	Y0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

- 2bit One-hot to Binary encoder
  - One-hot representation simplifies logic design i.e. only 1 bit is ever set.
    - Identify when output is a logic 1 then join active inputs with OR gates.

Encoder



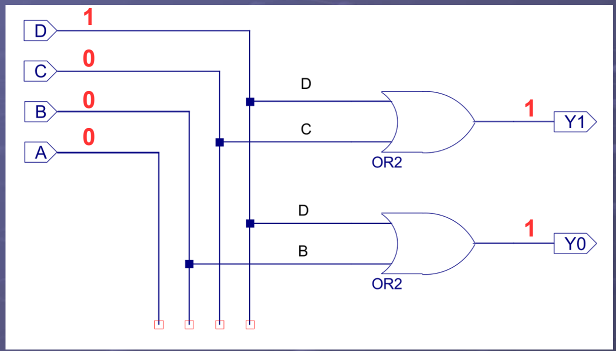


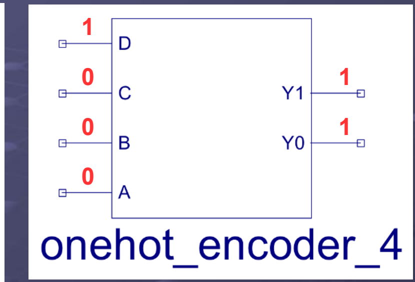
D	C	B	A	Y1	Y0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

- 2bit One-hot to Binary encoder
  - One-hot representation simplifies logic design i.e. only 1 bit is ever set.
    - Identify when output is a logic 1 then join active inputs with OR gates.

Slide 33

Encoder





D	C	B	A	Y1	Y0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

- 2bit One-hot to Binary encoder
  - One-hot representation simplifies logic design i.e. only 1 bit is ever set.
    - Identify when output is a logic 1 then join active inputs with OR gates.

Slide 34

Encoder / Decoder

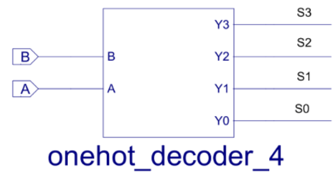
Which of the following is a valid one-hot value?
A) 00000000 B) 01000000 C) 10000010

An instruction is represented within a computer by a two bit binary number:

A B
0 1 = Add
1 0 = Multiply

A B
0 0 = Subtract
1 1 = Divide

This two bit code is decoded using the `onehot_decoder_4` component to produce control signals `S0`, `S1`, `S2` and `S3`. Complete tge truth table below, identifying what instruction is being processed.

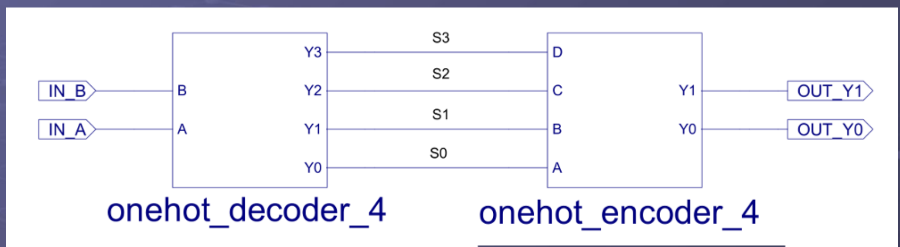


S3	S2	S1	S0	Instruction
0	0	0	1	?
1	0	0	0	?
0	0	1	0	?
0	1	0	0	?
0	1	0	1	?

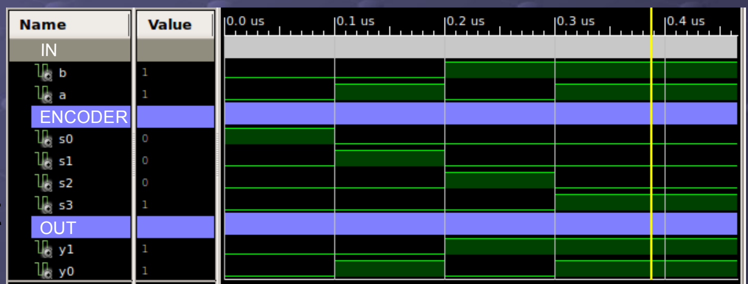
- Quick quizzz

Slide 35

Example : onehot_encoder.zip

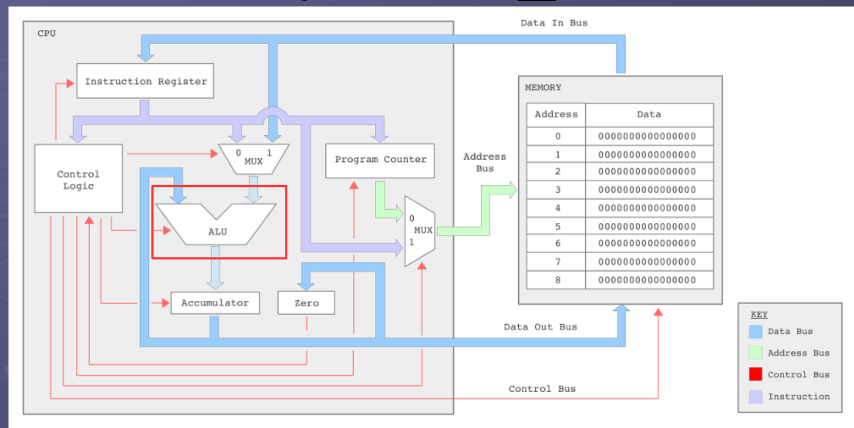


- Simulation
  - 2 bit input A, B
  - 2 bit output Y1, Y0



Slide 36

SimpleCPU_v1a



- Block diagram
  - ALU : a core requirement of any computer is to process data i.e. the Arithmetic and Logic unit, at its heart is the ADDER.

Key skills : working in base 2

```
  0110101        53
 +0011100       +28
 --------       ----
        1
 --------       ----
```

- Adding two binary numbers : 53 + 28
  - Positive, integer

Key skills : working in base 2

```
  0110101        53
 +0011100       +28
 --------       ----
       01
 --------       ----
```

- Adding two binary numbers : 53 + 28
  - Positive, integer

Key skills : working in base 2

```
  0110101        53
 +0011100       +28
 --------       ----
      001
 --------       ----
     1
```

- Adding two binary numbers : 53 + 28
  - Positive, integer

Key skills : working in base 2

```
  0110101        53
 +0011100       +28
 --------       ----
     0001
 --------       ----
    11
```

- Adding two binary numbers : 53 + 28
  - Positive, integer

Slide 41

Key skills : working in base 2

```
  0110101        53
 +0011100       +28
 --------       ---
    10001
 --------       ---
   111
```

- Adding two binary numbers : 53 + 28
  - Positive, integer

Slide 42

Key skills : working in base 2

```
  0110101        53
 +0011100       +28
 --------       ---
   010001
 --------       ---
  1111
```

- Adding two binary numbers : 53 + 28
  - Positive, integer

Slide 43

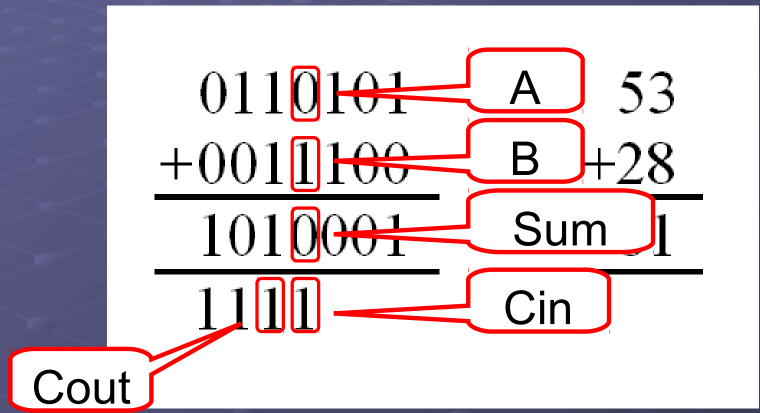
Key skills : working in base 2

```
  0110101        53
 +0011100       +28
 --------       ---
  1010001        81
 --------       ---
  1111
```

- Adding two binary numbers : 53 + 28
  - Positive, integer

Slide 44

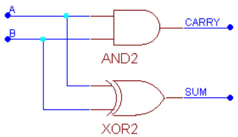
Binary addition



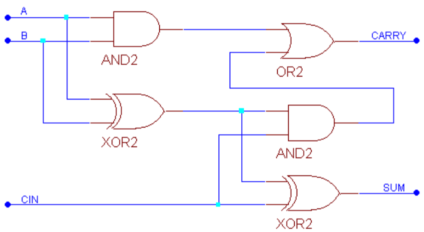
- Adding two binary numbers : 53 + 28
  - Positive, integer

Binary addition

A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

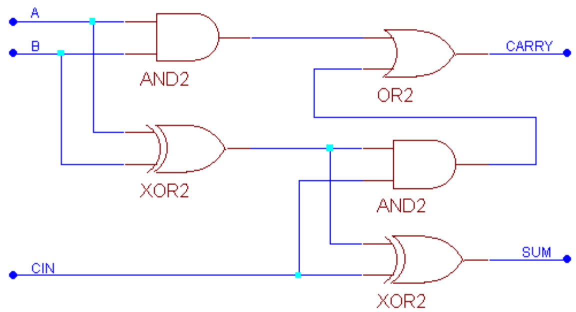


A	B	Cin	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

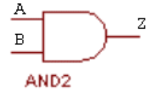


- Half and full adder
  - Basic components can be combined into larger circuits

Binary addition



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

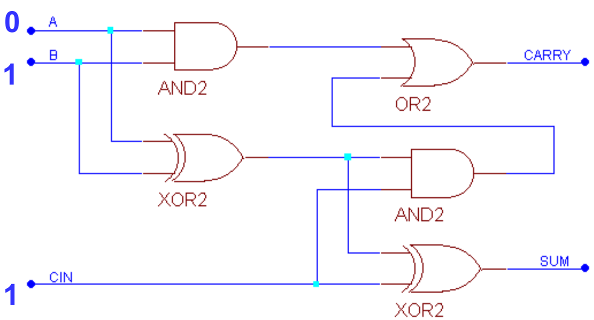


A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

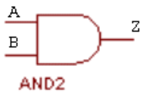
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

- Full Adder operation

Binary addition



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

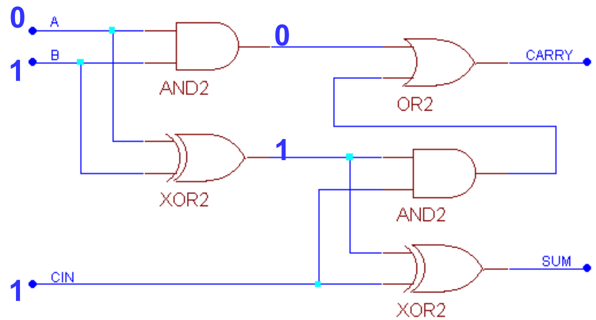


A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

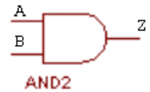
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

- Full Adder operation
  - A=0, B=1, C=1

Binary addition



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1



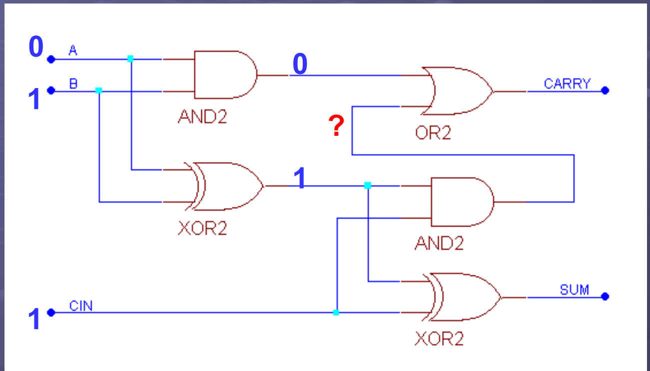
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

- Full Adder operation
  - Update outputs with stable values

Slide 49

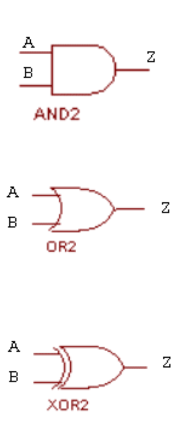
Binary addition



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

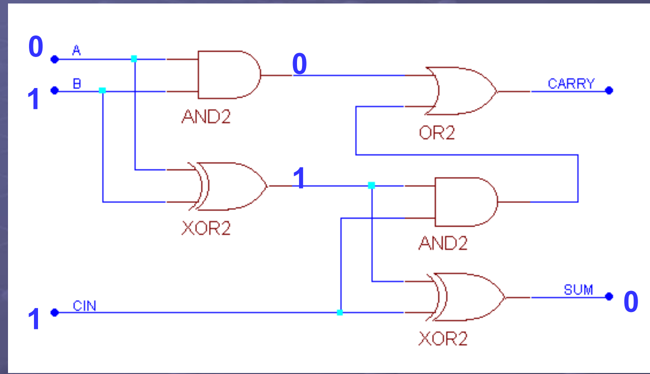
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0



- Full Adder operation
  - If input not known, trace signal back to source

Slide 50

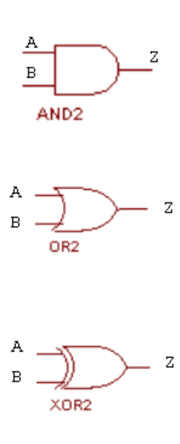
Binary addition



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

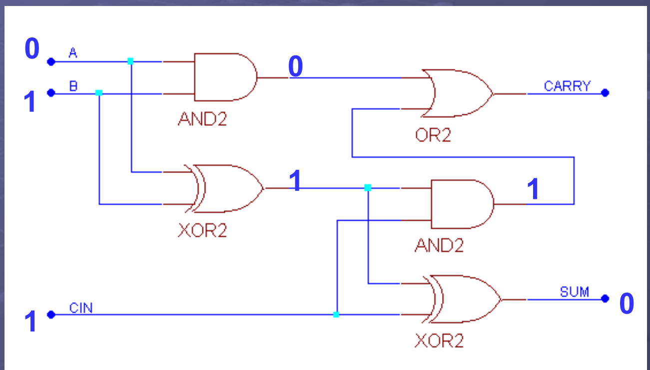
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0



- Full Adder operation
  - Update outputs with stable values

Slide 51

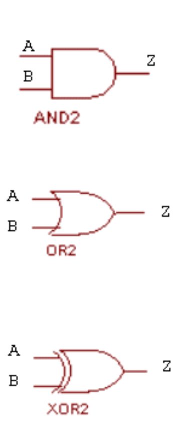
Binary addition



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

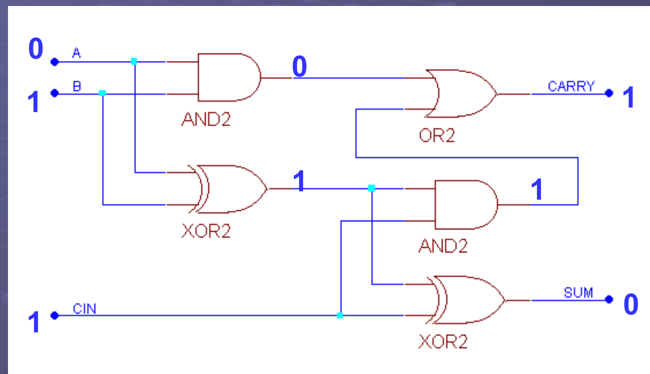
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0



- Full Adder operation
  - Update outputs with stable values

Slide 52

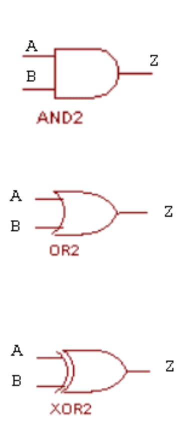
Binary addition



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

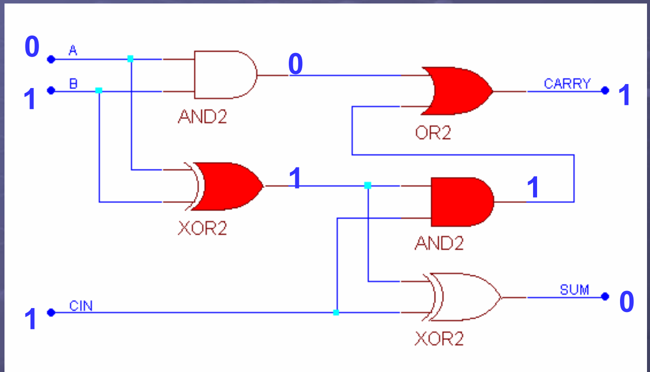
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0



- Full Adder operation
  - A=0, B=1, C=1
  - Sum=0, Carry=1

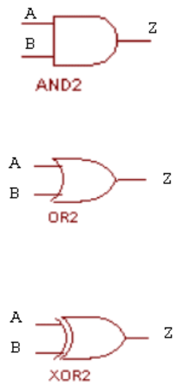
Binary addition



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

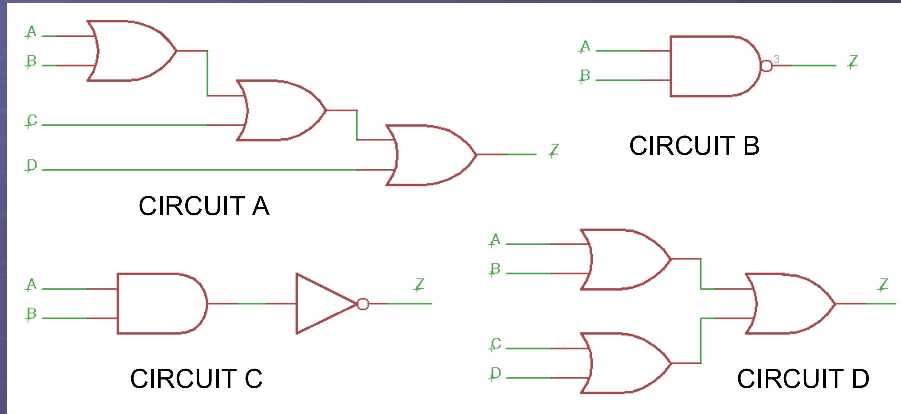
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0



- Important limitation : critical path
  - Worst case delay path, a signal needs to travel through three gate to produce a stable output.

Binary addition



- Quick Quizzz
  - Which rank these circuits in order of critical path delay, quickest to slowest.

Summary

- Key concepts :
  - Fundamental building blocks of a computer:
    - Multiplexer (bit and byte), Encoder, Decoder, Adder ...
  - Binary encodings
    - Binary, BCD, One-hot, Gray code …
      - Link : https://en.wikipedia.org/wiki/Gray_code
  - Binary arithmetic
    - Half adder, Full adder, Ripple adder, Carry, Overflow.
    - Hardware limitations : Critical Path Delay (CPD)
      - Each logic gate will take some time to update its output for a change on its input i.e. propagation delay.
  - Operation of simple combinatorial logic circuits