COMPUTER SYSTEMS AND ORGANIZATION

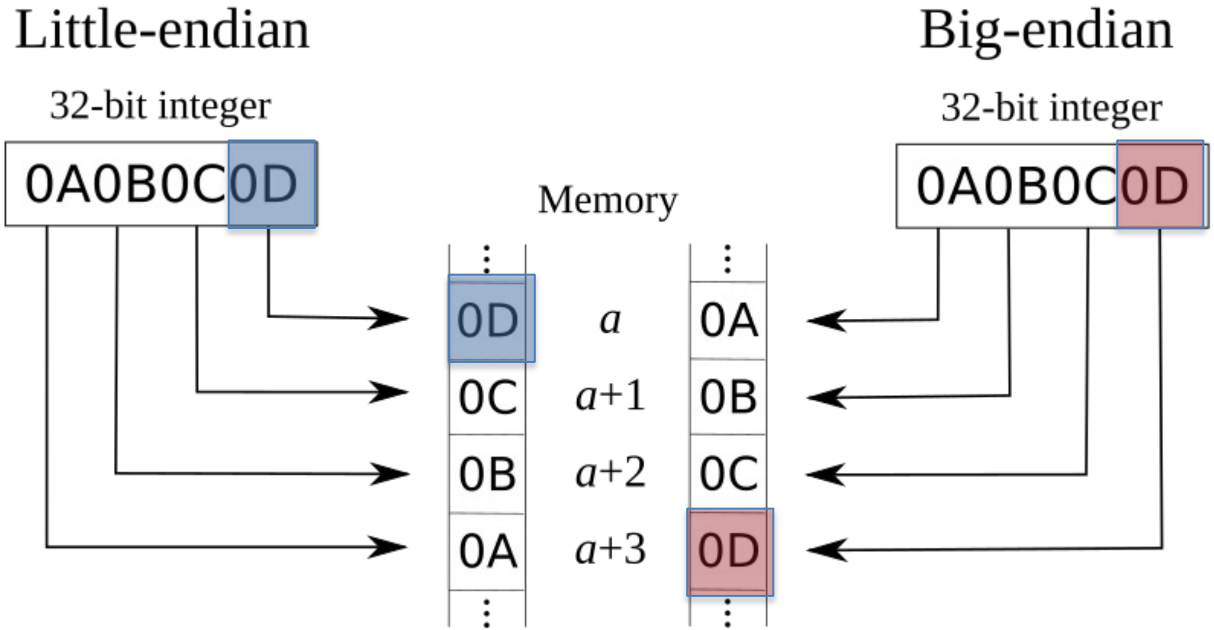
Adders

Daniel Graham

University of Virginia | Engineering

REVIEW

ENDIANNESS



Little-endian
32-bit integer
0A0B0C0D
Memory
0D
0C
0B
0A
a
a+1
a+2
a+3
0A
0B
0C
0D
Big-endian
32-bit integer
0A0B0C0D

EXAM QUESTION FALL 2018

Question 13 [2 pt]: If the 32-bit number `0x12345678` is stored in **little-endian** at address `0x20`, what is the value of the byte at address `0x22`? Answer in hexadecimal.

Answer:

Question 14 [2 pt]: If you read the bytes [`fe`, `dc`] as an unsigned **big-endian** 16-bit number, what is that number? Answer in hexadecimal.

Answer:

EXAM QUESTION FALL 2018

Question 13 [2 pt]: If the 32-bit number `0x12345678` is stored in **little-endian** at address `0x20`, what is the value of the byte at address `0x22`? Answer in hexadecimal.

Answer:

0x34

Question 14 [2 pt]: If you read the bytes [`fe`, `dc`] as an unsigned **big-endian** 16-bit number, what is that number? Answer in hexadecimal.

Answer:

0xfedc

EXAM QUESTIONS FALL 2019

Suppose an array of two 32-bit values (`[0xabcdef01, 0x7645231]`) is stored at address `0x200`. What byte is stored at address `0x204`? Answer in hexadecimal.

Question 13 [2 pt]: (see above) Assume little-endian storage.

Answer:

Question 14 [1 pt]: (see above) Assume big-endian storage.

Answer:

Remember to draw a picture of how arrays are stored in memory.

(Remember to group second number into bytes)

EXAM QUESTIONS FALL 2019

Suppose an array of two 32-bit values (`[0xabcdef01, 0x7645231]`) is stored at address `0x200`. What byte is stored at address `0x204`? Answer in hexadecimal.

Question 13 [2 pt]: (see above) Assume little-endian storage.

Answer:

0x31

Question 14 [1 pt]: (see above) Assume big-endian storage.

Answer:

0x07

CONVERSION

0.1 x 2 = 0.2	**0**
0.2 x 2 = 0.4	**0**
0.4 x 2 = 0.8	**0**
0.8 x 2 = 1.6	**1**
0.6 x 2 = 1. 2	**1**
0.1 x 2 = 0.2	**0**

…… repeats …

Just like the 1/3 0.1 keeps repeating

0 01111011 1001100 11001100 1100110

123

$\frac{1}{2} + 1/2^4 + 1/2^5 + 1/2^8 + 1/2^9 + 1/2^{12} + 1/2^{13} + 1/2^{16} + 1/2^{17} + 1/2^{20} + 1/2^{21}$

$(-1)^s \times (1 + m) \times 2^{\text{exponent} - \text{bias}}$

$0.099999994039535522460937 5 = (-1)^0 \times (1 + \frac{1\,258\,291}{2\,097\,152}) \times 2^{123 - 127}$

No quite 0.1

UNIVERSITY of VIRGINIA | ENGINEERING

EXAM QUESTION

Page 5: Floating Point and Bitwise Operations

8. [9 points] Write the following binary number as an 8-bit floating point number assuming a 4-bit exponent value.

`+0.00001110011`

Answer

EXAM QUESTION

Page 5: Floating Point and Bitwise Operations

8. [9 points] Write the following binary number as an 8-bit floating point number assuming a 4-bit exponent value.

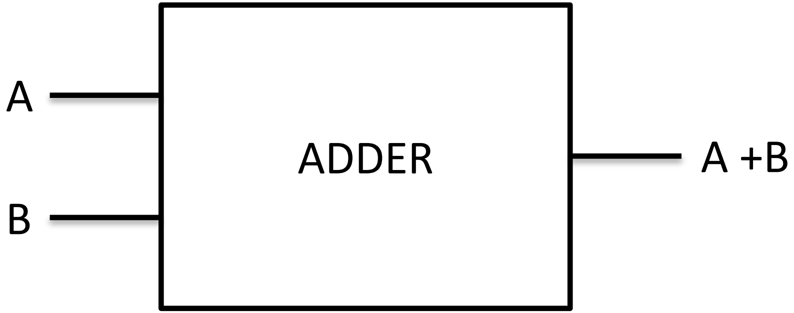
```
+0.00001110011
```

Answer
0 0010 110

TODAY'S LECTURE

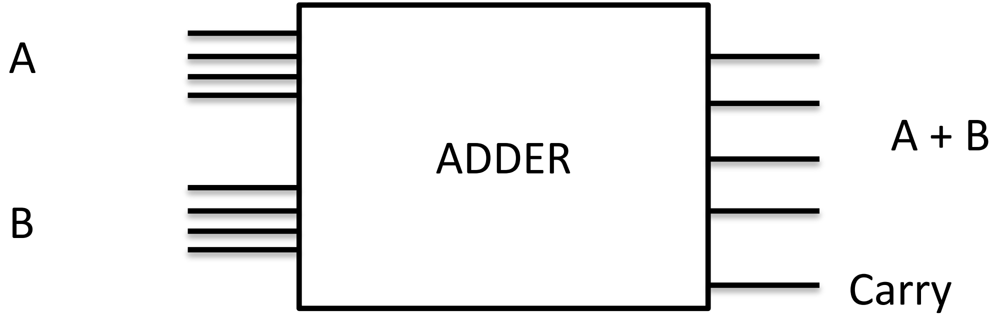
Contents

THE IDEA



UNIVERSITY of VIRGINIA | ENGINEERING

4-BIT ADDER



Great now let's build it with gates.

ADDING

```
1 1 1 1   ← Carries
  0 1 1 1
+ 1 0 1 1
  1 1 1 0
```

Let start by building a half adder something that just adds two bits.

Let's build a truth table.

A	B	A + B	C.out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

We can implement
A + B with an XOR gate
And the C.out (Carry out)
With an AND gate

ADDING

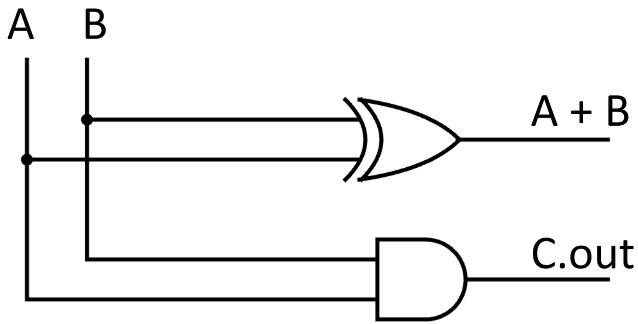
```
1 1 1 1   ← Carries
  0 1 1 1
+ 1 0 1 1
  -------
  1 1 1 0
```

Let start by building a half adder something that just adds two bits.

Let's build a truth table.

A	B	A + B	C.out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

We can implement
A + B with an XOR gate
And the C.out (Carry out)
With an AND gate

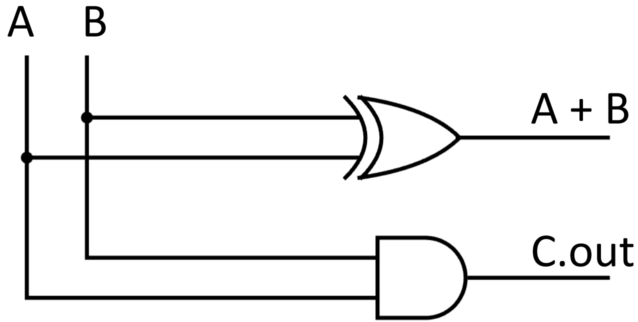


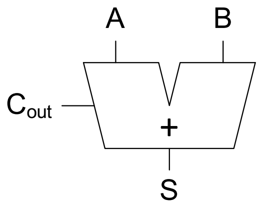
ADDING

Half Adder

```
1 1 1 1   ← Carries
  0 1 1 1
+ 1 0 1 1
  -------
  1 1 1 0
```

We can implement
A + B with an XOR gate
And the C.out (Carry out)
With an AND gate



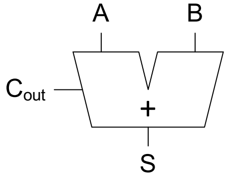


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = A \oplus B$

$C_{out} = AB$

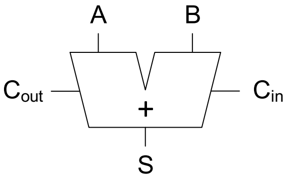
Half Adder



A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = A \oplus B$
$C_{out} = AB$

Full Adder

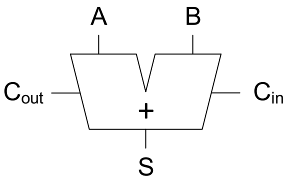


C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$S = A \oplus B \oplus C_{in}$
$C_{out} = AB + AC_{in} + BC_{in}$

```
1 1 1 1   ← Carries
  0 1 1 1
+ 1 0 1 1
  1 1 1 0
```

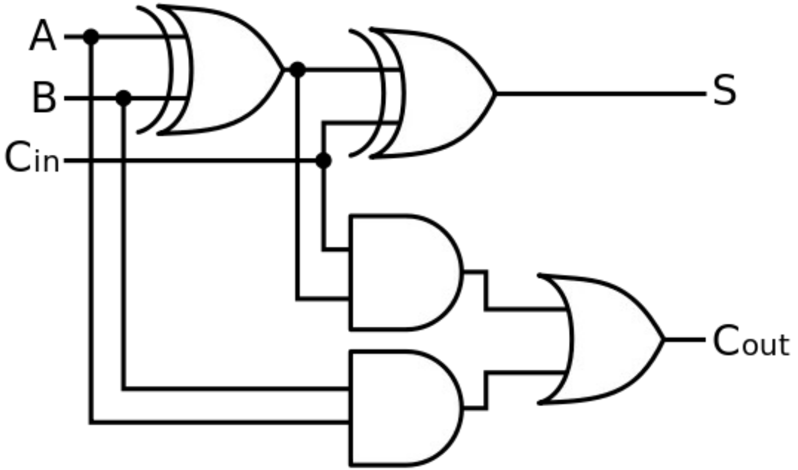
Full Adder



C_{in}	A	B	C_{out}	S
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

$S = A \oplus B \oplus C_{in}$
$C_{out} = AB + AC_{in} + BC_{in}$

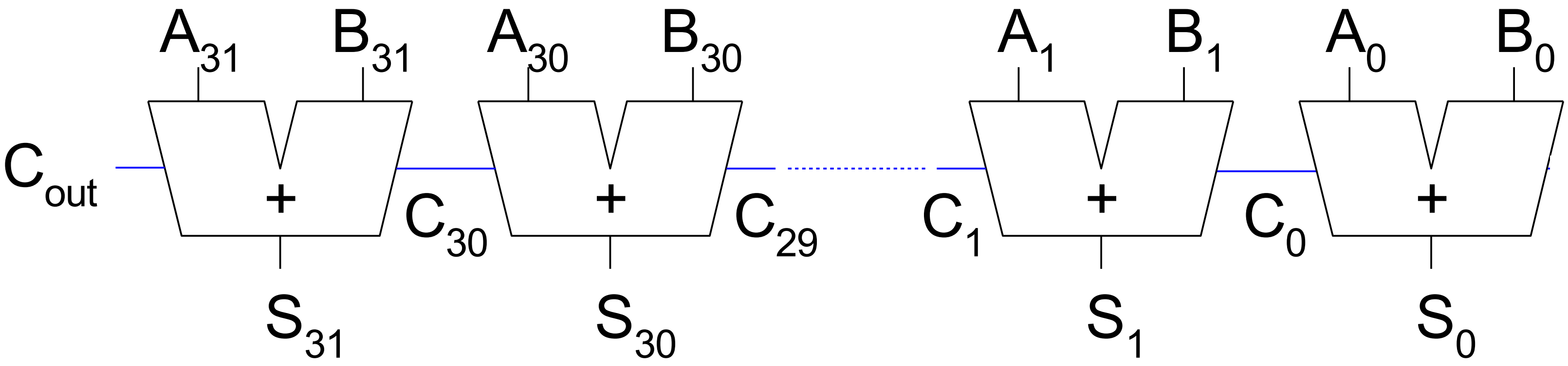
```
1 1 1 1   ← Carries
  0 1 1 1
+ 1 0 1 1
---------
  1 1 1 0
```

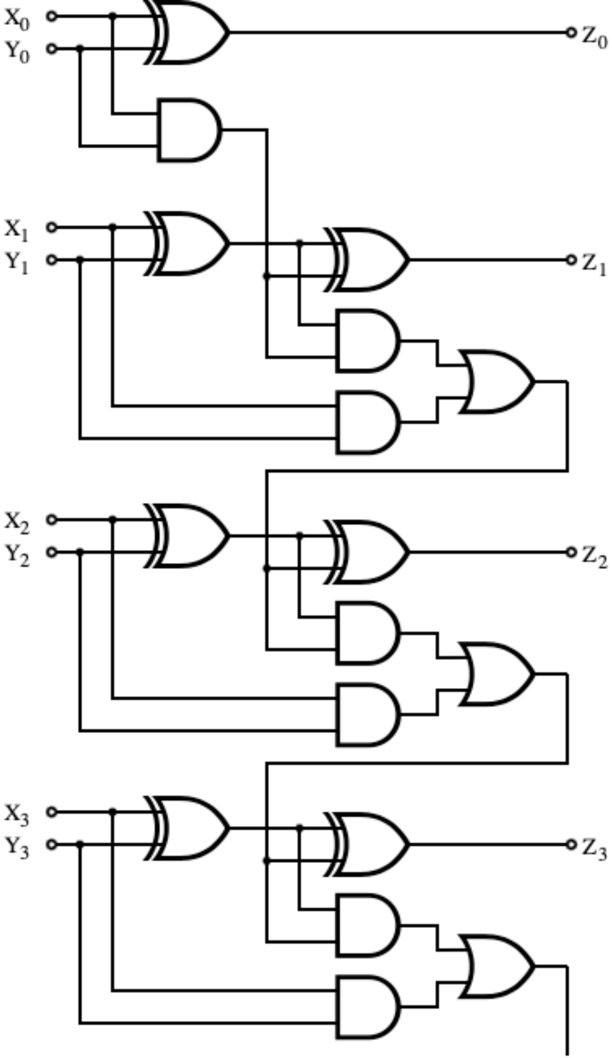


C.out has been rewritten to reduce the number of gates needed.

RIPPLE CARRY ADDER

Next let's build a full adder





RIPPLE CARRY ADDER

1 1 1 1 ← Carries

  0 1 1 1

+ 1 0 1 1

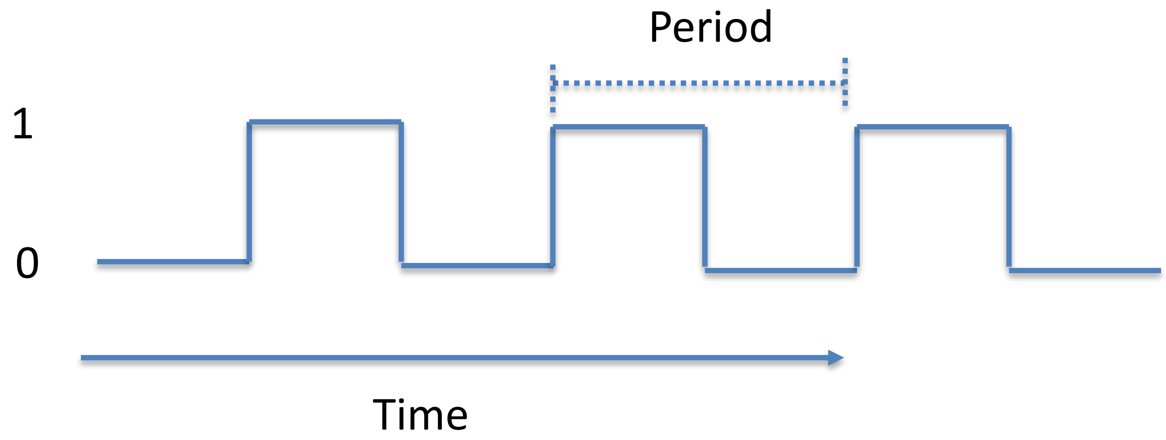
  1 1 1 0

CLOCKS

A clock is something that produces a periodic signal

Period is length of time for one clock cycle

Example



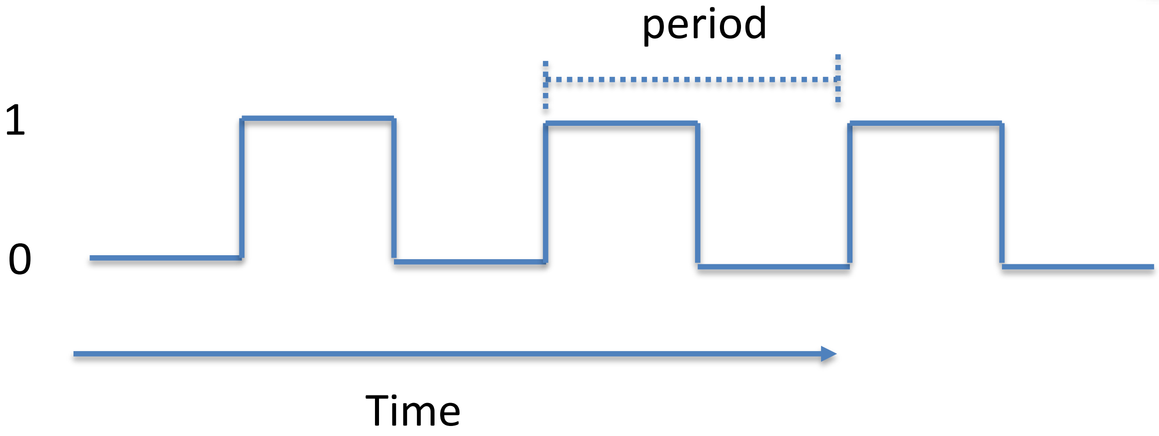
University of Virginia | Engineering

CLOCKS

Clock frequency Intel Core i-9 3.0GHz

Frequency = 1/ period

Period = 1/frequency

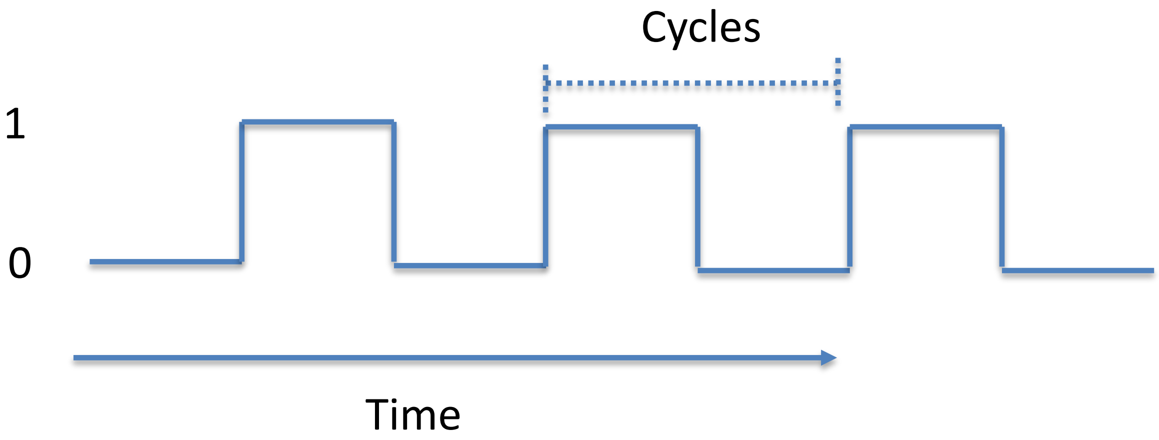


CLOCKS

Clock frequency Intel Core i-9

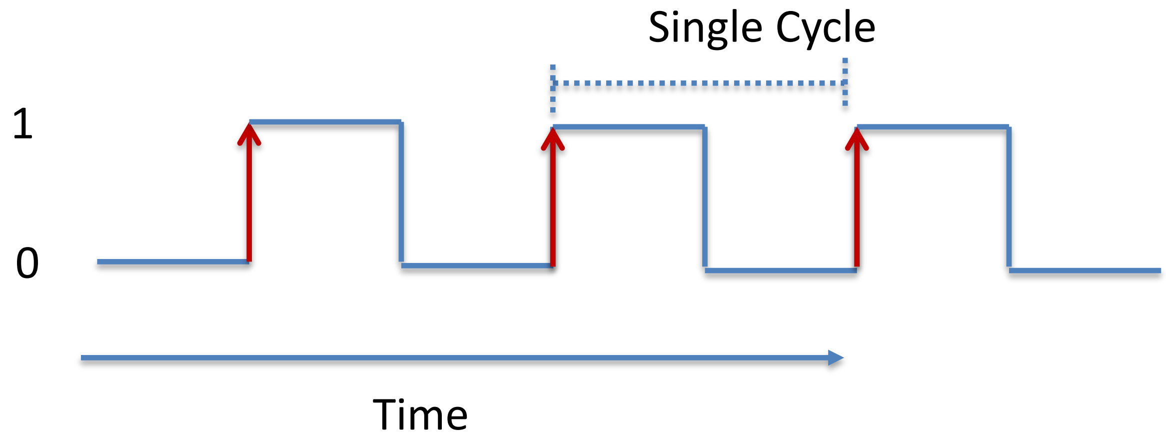
3.0GHz. = $3*10^9$ = 3,000,000,000 cycles per second

Thank is fast.



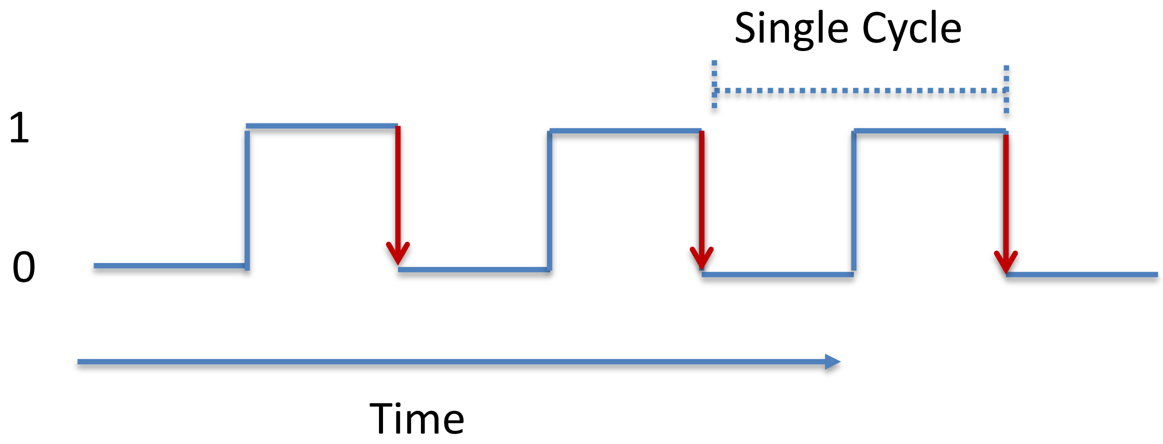
CLOCKS EDGES

Rising Edge



CLOCKS EDGES

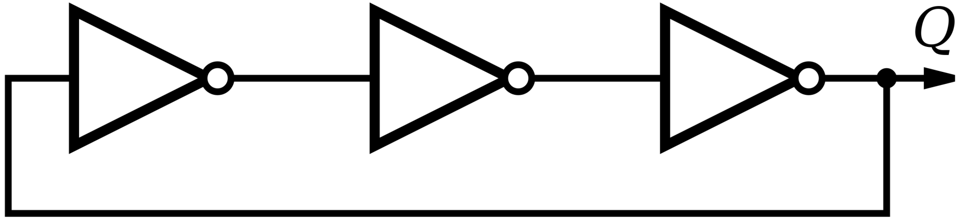
Falling Edge.



We will build a single cycle machine it will complete all the computation in a single cycle

USING RING OSCILLATORS TO GENERATE CLOCKS

A clock is something that produces a periodic signal



Let’s walk through an example assume that Q starts of as 0

Frequency = 1/(2*t*n)

Where t is time delay of an inverter and n is number of inverters

UNIVERSITY of VIRGINIA | ENGINEERING