Digital Logic

Logic

- LOGIC DESIGN
  - High level logical designs (programs) can be broken down into simple logical constructs
  - Boolean Algebra, Truth Tables and Electronic Logic Gates are all equivalent expressions
  - Expressions involve digital/binary/boolean logic
  - Gates are actual able to be implemented using high/low voltages (1/0)
- Logic
  - Combinational - no memory - output depends ONLY on current input
  - Sequential - includes state - memory

Logic

- Truth Table
  - Lists all possible input combinations
  - n inputs
  - 2^n entries - lists all relevant outputs
  - 2 inputs - 4 entries
  - 3 inputs, 8 entries, etc...
  - Can completely describe ANY combinational logic function
  - Tables for AND ,OR, NOT, NAND, NOR,XOR

Basic Components: Truth Tables

AND

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

OR

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

NOT

A	Out
0	1
1	0

NAND

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

NOR

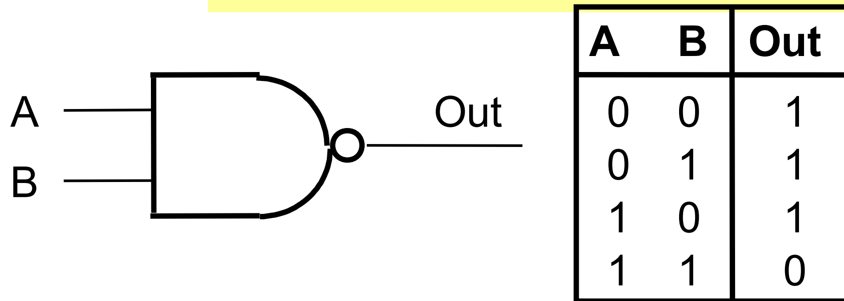
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

XOR

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

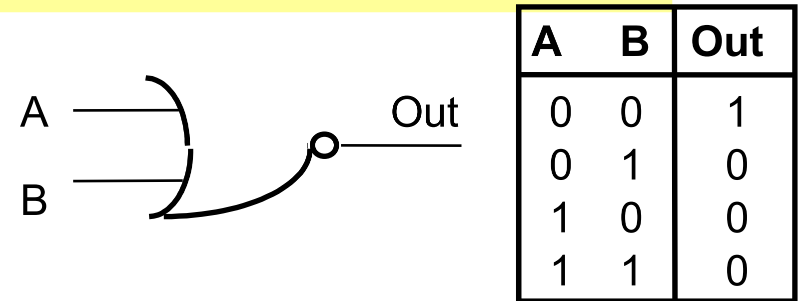
Basic Components: Logic Gates

NAND Gate



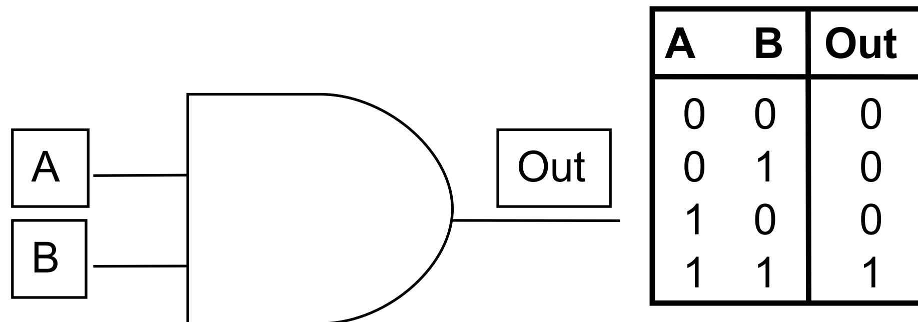
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate



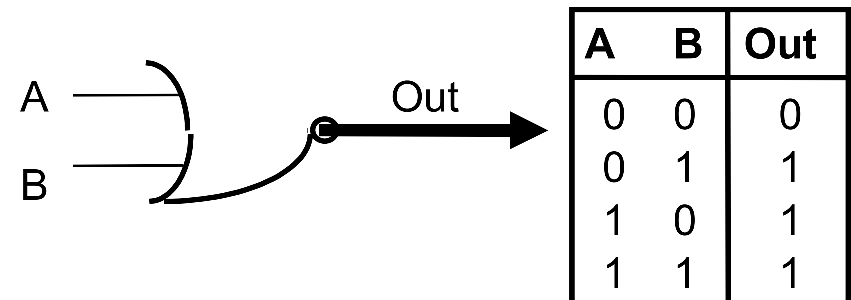
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

AND Gate



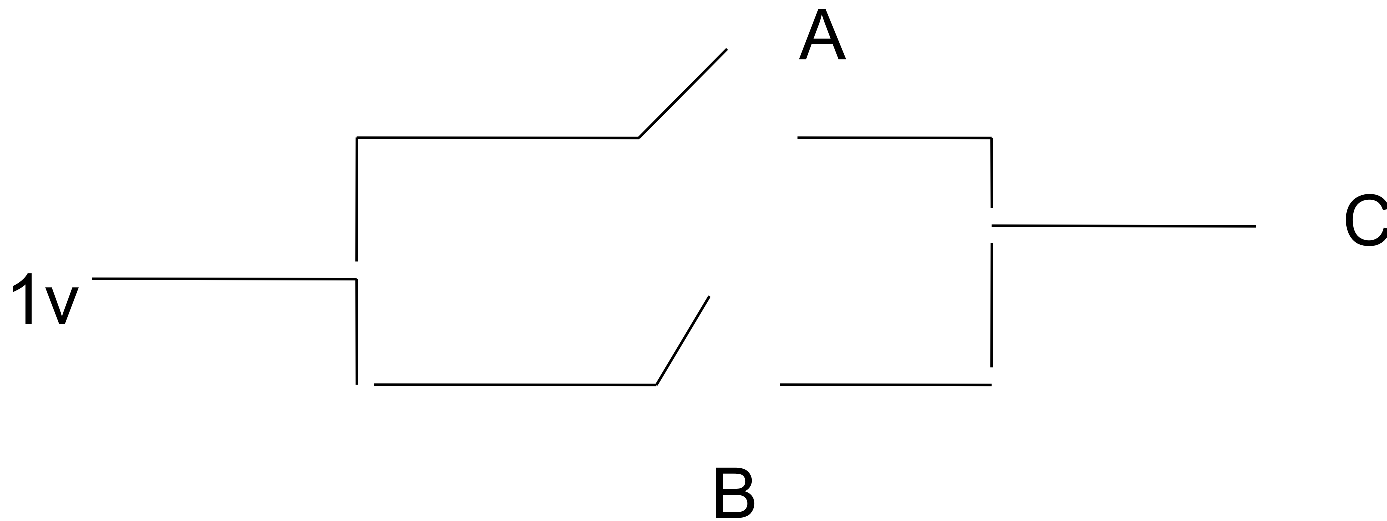
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate



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

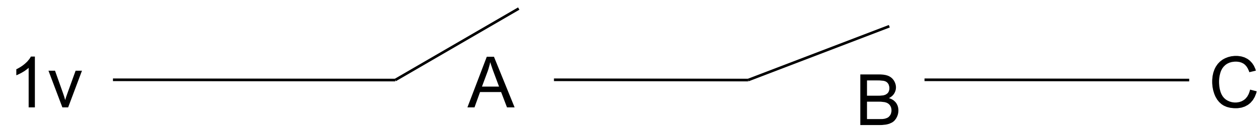
Basic Components: Gate



C= A ? B   what kind of gate??

Note: can have multiple inputs (C= A + B + D)

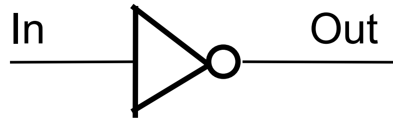
Basic Components: Gate



C= A ? B   what kind of gate??

Note: can have multiple inputs (C= A • B • D)

Basic Components:NOT



A	Out
0	1
1	0

NOT Gate

Logic

- Boolean Algebra
- Logic equations
- All variables = either 0 or 1
- 3 basic operators
- **AND, ●**
- **OR, +**
- **NOT , ¬**
- Can derive
- XOR, NAND, NOR

Logic

- COMBINATIONAL LOGIC
- Simple logical expressions
- Can Create an exclusive or from the 3 basic gates
  - (and, or, not)
  - Do it!
    - draw gates

XOR truth table

A	B	Out
0	0	0
0	1	**1**
1	0	**1**
1	1	0

Logic

- COMBINATIONAL LOGIC
- Simple logical expressions
- Can Create an exclusive or from the 3 basic gates
  - Using only the rows where result is 1
    - ((not A) and B) or (A and (not B))

Logic

- How did we get an XOR as
  - ((not A) and B) or (A and (not B)) ???

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

XOR

Look at the two rows where Out =1...
When is Out ==1??
1. When A is 0 (not A) and B is 1
   [we can write as: ((not A) and B)]
   OR
2. When A is 1 and B is 0 (not B)
   [we can write as: (A and (not B))]

so….

((not A) and B) or (A and (not B))

Logic

- Can find *any* equation for *any* truth table

```
ABC D
000 1   =(¬A • ¬B • ¬C)
001 0
010 1   =(¬A • B • ¬C)
011 0
100 0
101 0
110 1   =(A • B • ¬C)
111 0
```

Logic: Sum of Products

$(\neg A \cdot \neg B \cdot \neg C) + (\neg A \cdot B \cdot \neg C) + (A \cdot B \cdot \neg C)$

ABC	D	
000	1	$= (\neg A \cdot \neg B \cdot \neg C)$
001	0	
010	1	$= (\neg A \cdot B \cdot \neg C)$
011	0	
100	0	
101	0	
110	1	$= (A \cdot B \cdot \neg C)$
111	0	

Logic: Sum of Products

ABC	D
000	1
001	0
010	0
011	0
100	0
101	1
110	0
111	0

Create boolean expression for this truth table.

Create a logic diagram using gates.

Logic: Sum of Products

ABC	D
000	1
001	0
010	0
011	0
100	0
101	1
110	0
111	0

Create boolean expression for this truth table.

$(\neg A \bullet \neg B \bullet \neg C) + (A \bullet \neg B \bullet C)$

Create a logic diagram using gates.

Logic: XOR

- BreadBoard Software
  - Create an XOR
  - Need two and gates and an or gate

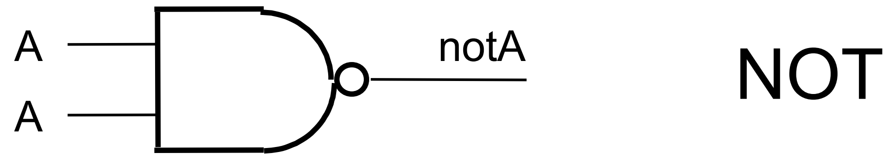
((not A) and B) or (A and (not B))

$((\neg A) \bullet B) + (A \bullet (\neg B))$

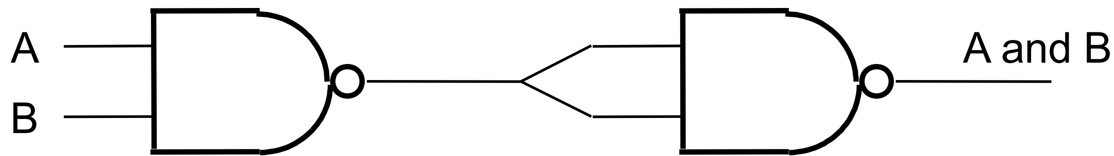
Due ______ by midnight

Logic: NAND

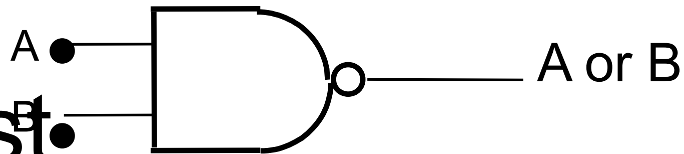
- NAND gates can implement ANY logic



NOT



AND



OR

- Cheap and fast

Logic: Half adder

- Binary addition
- A simple half adder
-   truth table
- A B C_o S
- 0 0 0 0
- 0 1 0 1
- 1 0 0 1
- 1 1 1 0

Create a boolean expression for a half adder using sum of products.

Carry

Sum

Logic: Half adder

- Binary addition
- A simple half adder
- truth table
- A B Co S
- 0 0 0 0
- 0 1 0 1
- 1 0 0 1
- 1 1 1 0

Create a boolean expression for a half adder using sum of products.

Carry

(A and B)

Sum

(A and (not B)) or ((not A) and B)

Logic: Half Adder

- Binary addition
- A simple half adder
- truth table

A	B	C_o	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Create a boolean expression for a half adder using sum of products.

Carry

(A and B)

Sum

(A and (not B)) or ((not A) and B)

Using gates, create a half adder!

If each gate is 20ns, what's the fastest clock?

Logic: Half Adder

- BreadBoard Software
- Create a Half Adder
  - Carry
  - (A and B)
  - Sum

Half.wbd

  - (A and (not B)) or ((not A) and B)
  - May use XOR gates for this
  - Due _________ @ 11

Logic: Full Adder

- Truth Table for a full adder (w/carry in)

ABCi	Co	Sum
000	0	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	1	1

Logic: Full Adder

- Truth Table for a full adder (w/carry in)

ABCi	Co	Sum
000	0	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	1	1

Create a boolean expression for Co and Sum

Logic: Full Adder

- Truth Table for a full adder (w/carry in)

ABCi	Co	Sum
000	0	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	1	1

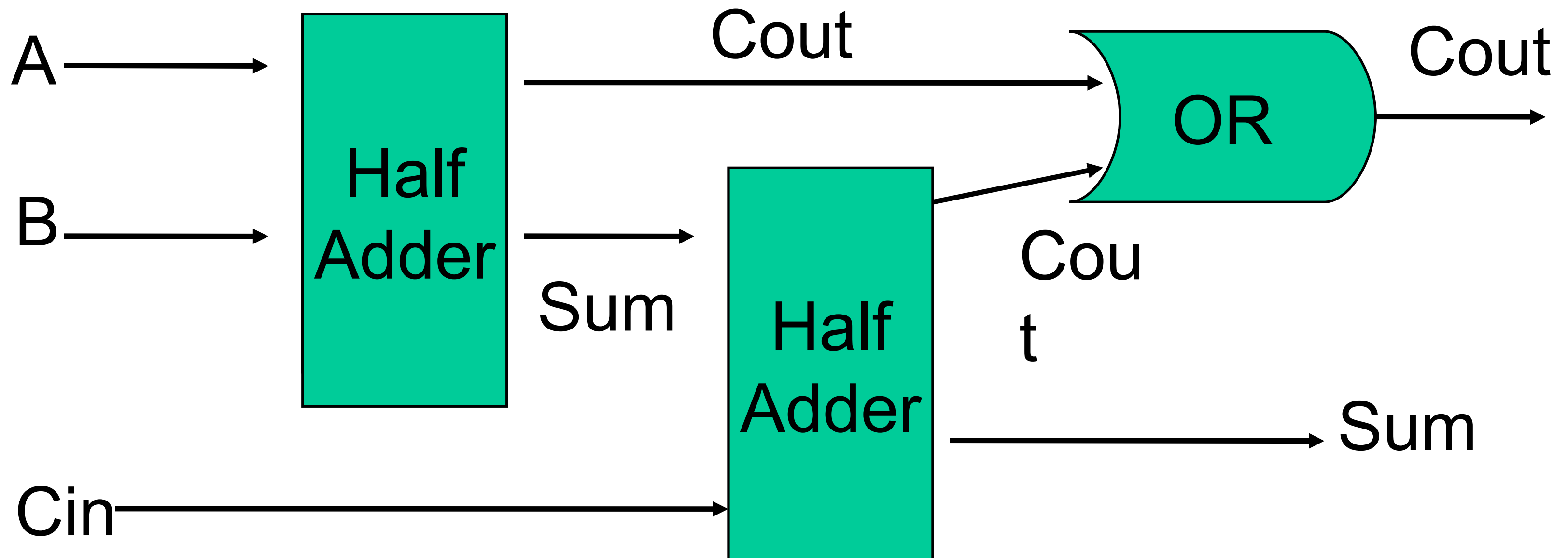
Create a boolean expression for Co and Sum

Now create a digital circuit that will implement that expression…

Is there a problem??

Logic

- Combining half adders to make a full adder



Logic: Full Adder

- BreadBoard Software
- Implement the 4 bit full Adder

Full.wbd

  - C0 is the Carry IN
  - C4 is the Carry OUT
  - 16 pins : pin 16 is Vcc, pin 8 is GRND
  - output to 7 segment display
  - detect unsigned overflow
  - Due __________ @ 10

Logic: Worksheet

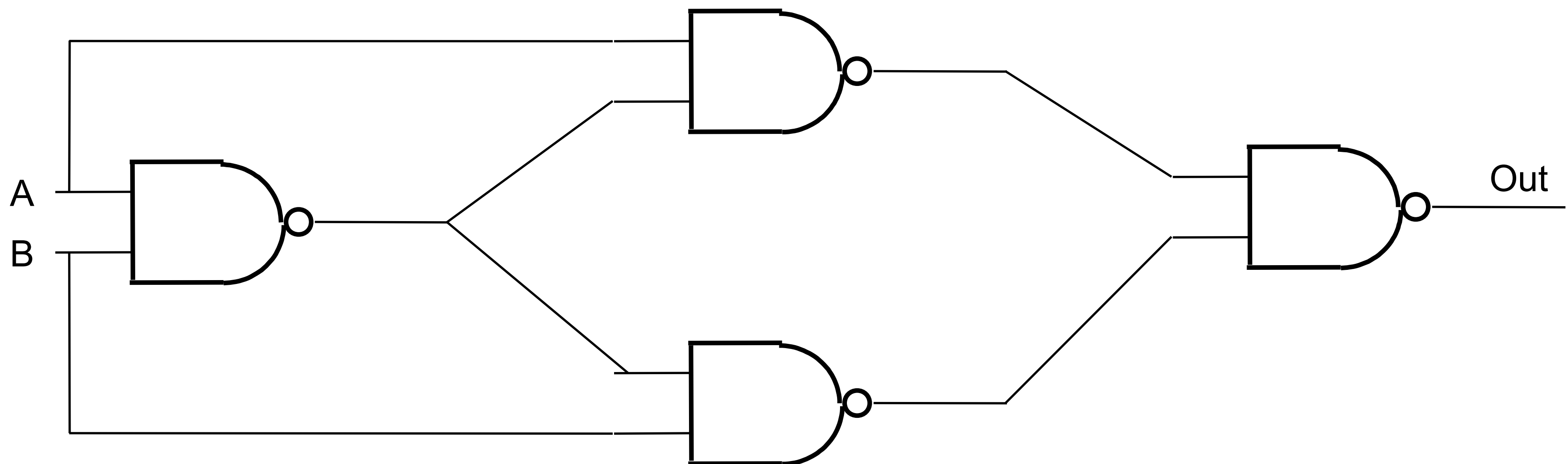
- Problem: Consider a logic function with three inputs: A, B, and C.
  Output D is true if at least one input is true
  Output E is true if exactly two inputs are true
  Output F is true only if all three inputs are true
- Show the truth table for these three functions.
- Show the Boolean equations for these three functions.
  Show an implementation consisting of inverters, AND, and OR gates.

Logic: Worksheet

- Worksheet
  - DeMorgan's Law states that
    - NOT (a OR b) = (NOT a) AND (NOT b)
    - Prove it with a truth table
    - Show equivalent digital logic using gates
  - DeMorgan's Law states that
    - a AND (b OR c) = (a AND b) OR (a AND c)
    - Prove it with a truth table
    - Show equivalent digital logic using gates

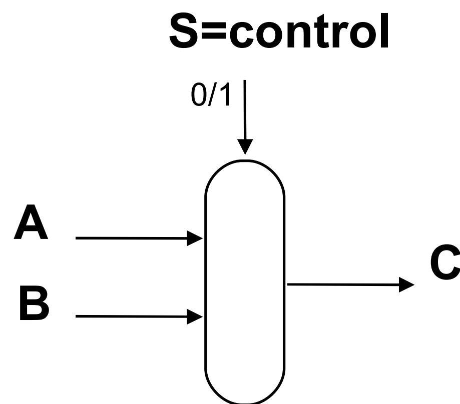
Logic: Worksheet

- Worksheet
  - create truth table and boolean expression



The Multiplexor

- Selects one of the inputs (A or B) to be the output, based on a control input

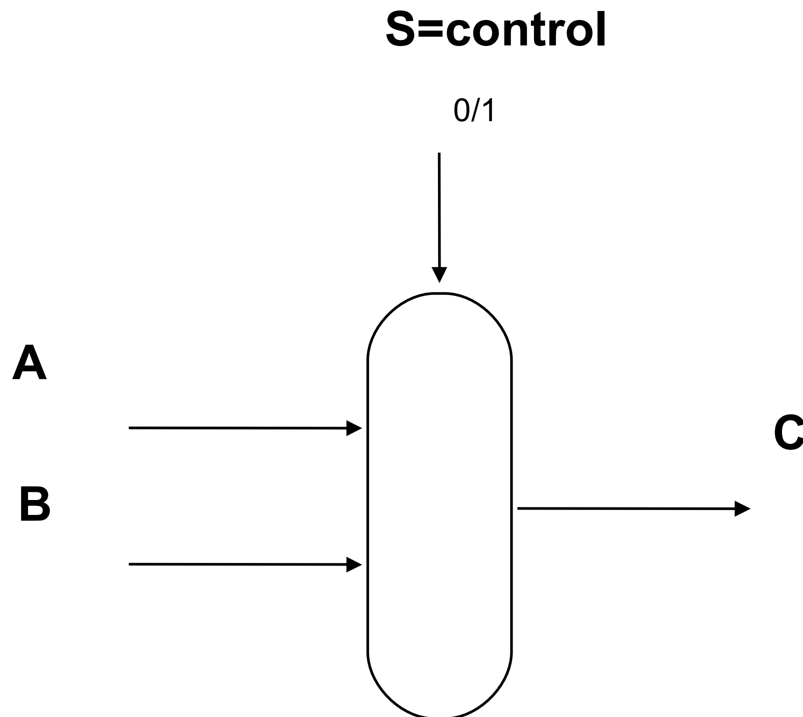


note: we call this a 2-input mux even though it has 3 inputs!

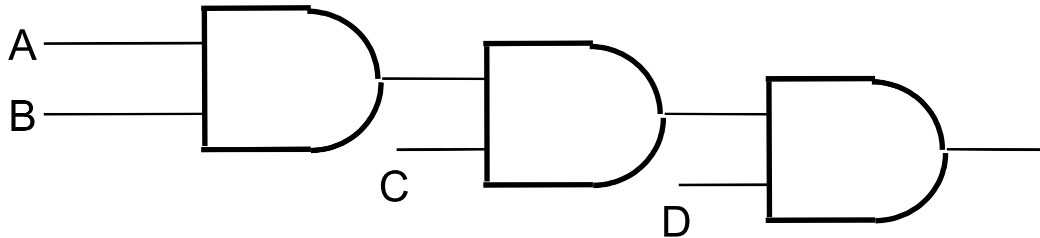
How many bits does S have to be to select 1 of 2 things? 3 things? 4? 5? 6? 8? 10?

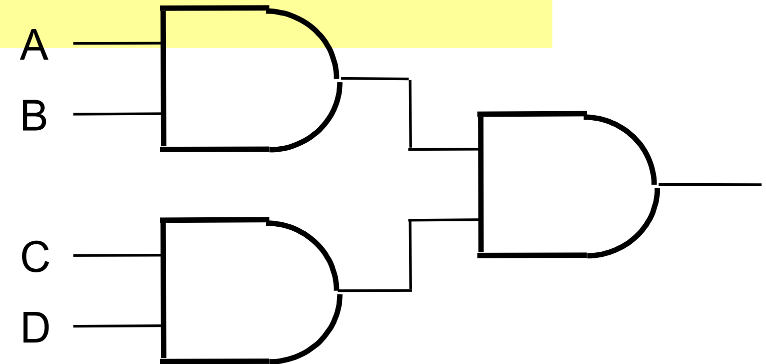
The Multiplexor

- If A = 01101111 and B= 00000001 and S =1 what will C be??



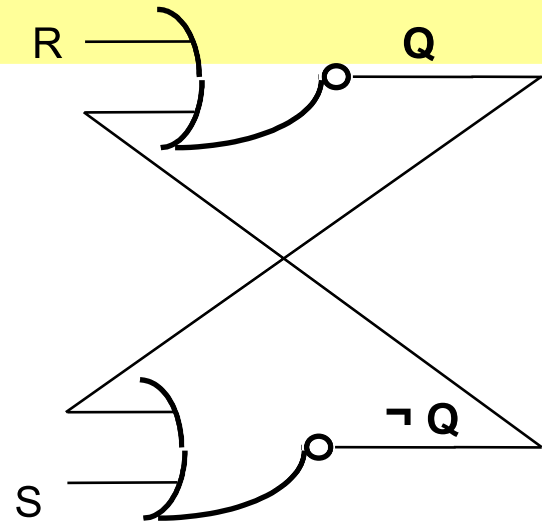
Tricks to Reduce Cycle Time





- Reduce the number of gate levels
- Faster Gates

S-R Latch: Memory



NOR

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

- Sequential Logic
- When neither R or S are asserted then the previous value is stored
- R=1 (clear), then Q=0 / S=1(set), then Q=1

Time

Execution Time

- Elapsed Time
  - counts everything *(disk and memory accesses, I/O , other programs executing, etc.)*
  - a useful number, but often not good for comparison purposes
- CPU time
  - doesn't count I/O or time spent running other programs
  - can be broken up into *system CPU time*, and *user CPU time*
- Our focus: *user CPU time*
  - time spent executing the lines of code that are "in" our program
  - not OS overhead - not other jobs while we wait
  - how do we measure??

Clock Cycles

- How do we measure time?
  - Clock cycle
    - time can be expressed as a series of clock ticks
    - a clock is the sequencing, stepping mechanism which is the smallest unit in which a hardware event can take place
    - each tick of the clock is called a CLOCK CYCLE or period
    - a cycle is measured as some fraction of a second (1 us or 100 ns)

Clock Cycles

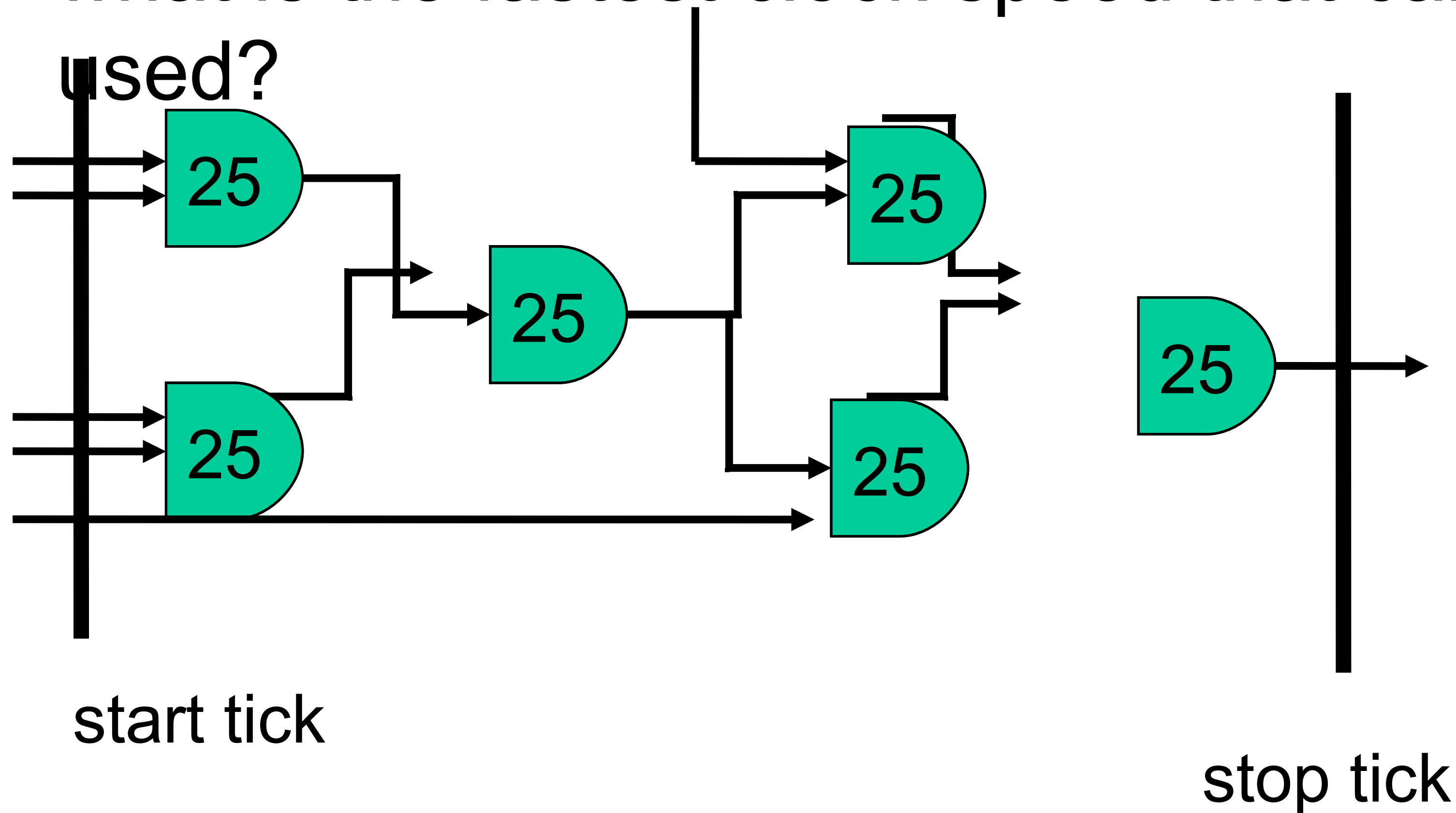
- CLOCK RATE measures how many times a clock ticks *per second*
  - a clock cycle is measured as the inverse of the clock rate
  - 1 hz is a single sine wave taking 1 second
  - if cycle = 1 second, rate = 1 hz
- A 100 hz clock is 100 waves in a single second with each cycle taking .01 sec .
  - So, cycle=.01, rate = 100 hz
- A 1 megahertz (1 mhz) clock ticks a million times per second
  - 1 million hz is a million waves in a sec each taking .000001 or 1 microsecond

Clock Cycles

- .000 000 000 000
- milli micro nano pico
- A 1 mhz clock has a 1/1million cycle = .000 001
- A 10 mhz clock has a 1/10million cycle = .000 000 100
- A 100 mhz clock has a 1/100million cycle = .000 000 010
- Why slow clocks?
  - electricity takes real time to travel
  - gates take real time to traverse
  - must allow signals to be stable in the slowest circuit
  - can't tick until outputs of one circuit are stable and can act as the inputs to the next

Clock Cycles

- If each AND gate takes 25ns to traverse, what is the fastest clock speed that can be used?



Clock Cycles

- Summary:
- Cycle time = length of a single clock tick = seconds per cycle (100 Hz = .001 cycle)
- Clock rate (frequency) = cycles per second (100 Hz. = 100 cycle/sec)
- Clock rate is INVERSE of Clock cycle
- A 200 Mhz. clock has a cycle time of:
  - ?
- A machine with a cycle time of 100 ns
  - ?

How to Improve Performance

- Instead of reporting execution time in seconds, we often use cycles
- Clock “ticks” indicate when to start activities (one abstraction):
- So, to improve performance (everything else being equal) you can either:

  _________ the # of required cycles for a program, or

  _________ the clock cycle time or, said another way,

  _________ the clock rate.

Example

- Execution time of the application program ultimately it is simply the number of clock cycles for a program x cycle time

  example:

  1. a program takes 300 clock cycles to execute
  2. each cycle is 10 ns
  3. 300 x 10 ns = 3000 ns = 3 microseconds

  example:

  cycle time is a given for a system (486/33mhz)

  how do you find number of clock cycles??

  1. find the number of instructions executed
  2. find the average clock cycles per instruction **(CPI**)
  3. time = CPI x (cycle time) x (# of instructions)

Example

Now that we understand cycles

- A given program will require
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds

Example

Now that we understand cycles

- We have a vocabulary that relates these quantities:
  - cycle time (seconds per cycle)
  - clock rate (cycles per second)
  - CPI (cycles per instruction)

    a floating point intensive application might have a higher CPI
  - MIPS (millions of instructions per second)

    this would be higher for a program using simple instructions

QUIZ

- 3 inputs: A, B, C
  - There are 2 outputs: D, E
  - D is True whenever only A & B are True
  - E is True whenever C is False
- Create a truth table
- Create a boolean expression using "sum of products"
- Create a logic gate diagram directly from the boolean expression (And, Or, Not)

Don't Simplify Anything
but MAY use 3 input gates

CPI Example

- Suppose we have two implementations of the same instruction set architecture (ISA).
- For some program with 100 instructions:
  - Machine A has a clock cycle time of 10 ns. and a CPI of 2.0
  - Machine B has a clock cycle time of 20 ns. and a CPI of 1.2
  - What machine is faster for this program, and by how much?
  - *If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?*

Worksheet

CIS 350 Architecture Worksheet #1 Performance

1. What is the clock speed of a clock with a period of 10 ns? 20 ns? 100 ns?

2. A clock running at the speed of 25 Mhz has a cycle of? 33 Mhz? 66 Mhz?

3. The execution time of a program depends on three things: what are those three things?

Worksheet

CIS 350 Architecture Worksheet #1 Performance

1. What is the clock speed of a clock with a period of 10 ns? 20 ns? 100 ns?

100MHz, 50 MHz, 10 MHz

2. A clock running at the speed of 25 Mhz has a cycle of? 33 Mhz? 66 Mhz?

40 ns, 30.30ns, 15.15 ns

3. The execution time of a program depends on three things: what are those three things?

instructions, cycles per instruction, cycle time

Worksheet

CIS 350 Architecture Worksheet #1 Performance

4. A benchmark with 100,000 instructions is being run on your new CPU. To have the fastest performance on this benchmark do you choose:

a. a CPI of 1.7 and a clock cycle of 19 ns

b. a CPI of 2.4 and a clock speed of 75 Mhz

5. Draw a diagram of a length of one second showing a clock speed of 5 hz. Label the length of each cycle. Calculate the length of a program with 100 instructions and a CPI of 1.

Worksheet

CIS 350 Architecture Worksheet #1 Performance

4. A benchmark with 100,000 instructions is being run on your new CPU. To have the fastest performance on this benchmark do you choose:

a. a CPI of 1.7 and a clock cycle of 19 ns

1.7 x 19ns x 100,000=3,230,000ns=3,230us=3.23ms

b. a CPI of 2.4 and a clock speed of 75 Mhz

2.4 x 13ns x 100,000=3,12,000ns=3,120us=3.12ms

5. Draw a diagram of a length of one second showing a clock speed of 5 hz. Label the length of each cycle. Calculate the length of a program with 100 instructions and a CPI of 1. cycle =.2 x 100 x 1= 20sec

of Instructions Example

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: *Class A, Class B, and Class C that require one, two, and three cycles respectively.*
- The first code sequence has **5 instructions**: <u>2 of A, 1 of B, and 2 of C</u>
- The second sequence has **6 instructions**: <u>4 of A, 1 of B, and 1 of C.</u>
- Which sequence will be faster? How much?
  What is the CPI for each sequence?

of Instructions Example

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: *Class A, Class B, and Class C that require one, two, and three cycles respectively.*
- The first code sequence has **5 instructions**: 2 of A, 1 of B, and 2 of C (2+2+6 = 10 cycles / 5 inst = 2.0 cpi)
- The second sequence has **6 instructions**: 4 of A, 1 of B, and 1 of C. (4+2+3 = 9 cycles / 6inst = 1.5 cpi)
- Which sequence will be faster? How much?
  What is the CPI for each sequence?

MIPS example

- Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.

  The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

  The second compiler's code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

- Which sequence will be faster according to MIPS?
- Which sequence will be faster according to execution time?

MIPS example

- Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.
- The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions. [10m cycles (faster) 7m inst
- The second compiler's code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions. [15m cycles (slower) 12m inst (but more inst. over given time)
- Which sequence will be faster according to MIPS?
- Which sequence will be faster according to execution time?

Amdahl's Law

Execution Time After Improvement =
Execution Time Unaffected +(Execution Time Affected / Amount of Improvement)

- Example: "Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?"
- How about 5 times faster?
- *Principle: Make the common case fast*