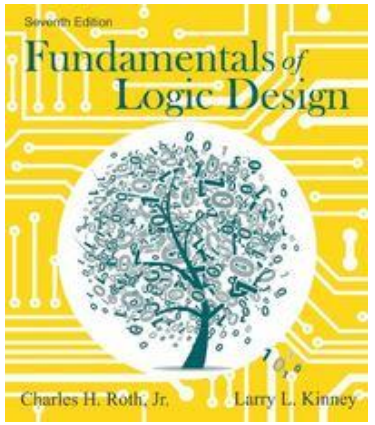
Digital Logic
PhD Qualifying Exam
Review Session

November, 2017

Quincy FLINT

Resources



- Textbooks (*I don't think you need one*)
  - Fundamentals of Logic Design – Roth

- UF Past Practice Exams
  - https://www.ece.ufl.edu/content/phd-written-qualifying-exam-questions

- UF Exam Study Guide
  - https://www.ece.ufl.edu/sites/default/files/pictures/DigitalLogic.pdf

- Exam Registration
  - https://gradadmissions.ece.ufl.edu/srs-servlet/examRegistration/phd

Important Dates

- **12/01/2017** – Qualifying exam registration closes (Friday, 5 pm)
- **12/??/2017** – Potential 2nd review session
- **01/??/2018** – Final review session
- **01/20/2018** – Qualifying exam (Saturday, time TBD)

Boolean Algebra

- **Algebra with 0's and 1's**
  - X + 0 = X
  - X + 1 = 1
  - X * 1 = X
  - X * 0 = 0
- **Idempotent Laws**
  - X + X = X
  - X * X = X
- **Complement Laws**
  - X + /X = 1
  - X * /X = 0

Boolean Algebra

- **Dual**:
  - 1 → 0
  - + → ×

  X + 0 = X → X * 1 = X

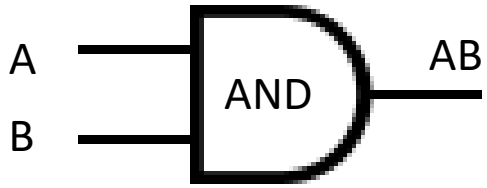
  X + 1 = 1 → X * 0 = 0

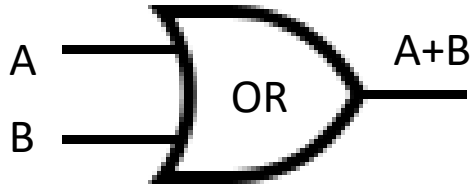
  X + /X = 1 → X * /X = 0

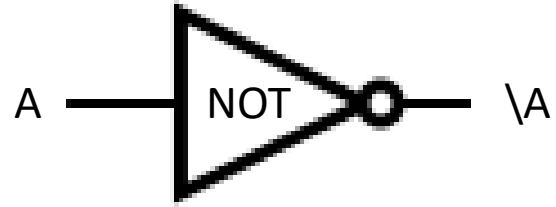
- **DeMorgan's Laws**
  - NOT(X+Y+Z) = NOT(X) * NOT(Y) * NOT(Z)
  - NOT(X*Y*Z) = NOT(X) + NOT(Y) + NOT(Z)
- **Consensus Theorem**
  - XY + **YZ** + \XZ = XY + \XZ

  Proof - example

Logic Gates





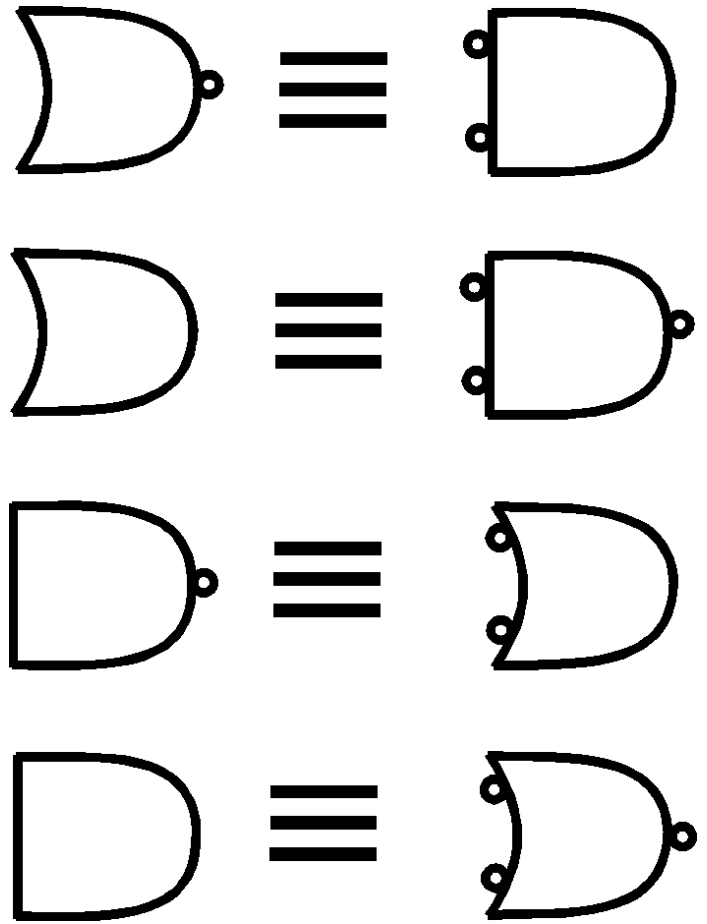


INPUT		**OUTPUT**		
A	B	A **AND** B	A **OR** B	**NOT** A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

More Logic Gates

- **Active High**
  - High Voltage (1) is TRUE
- **Active Low**
  - Low Voltage (0) is TRUE

- Active Low to bubble = match
- Active High to bubble = mismatch
- Bubbles do not change logic



Number Systems

- **Base 10**: $541 = 5\times10^2 + 4\times10^1 + 1\times10^0$

  $= 5\times100 + 4\times10 + 1\times1$

- **Base 2**: $0101 = 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

  $= 0\times8 + 1\times4 + 0\times2 + 1\times1$

Conversion between systems

- **Decimal to Binary**
  1. Brute Force -- “Count it out”
  2. Divide by 2 – remainder becomes binary (least to most significant)

- **Decimal to Hex**
  1. Convert to Binary then groups of 4 bits
  2. Divide by 16 -- remainder becomes hex (least to most significant)

- **Decimal to Octal**
  1. Convert to Binary then groups of 3 bits
  2. Divide by 8 -- remainder becomes octal (least to most significant)

Signed Number Representations

- **Signed Magnitude**:
  - MSB gives sign
  - 1000 0101 = -5

- **1's Complement**:
  - if MSB is 1 – flip bits and apply minus sign
  - If MSB is 0 – do nothing, positive
  - 1111 1100 (flipped = 0000 0011) = -3

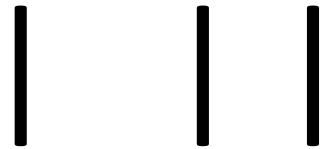
- **2's Complement**:
  - if MSB is 1 – flip bits, add 1, and apply minus sign
  - 1111 1101 (flipped + 1 = 0000 0011) = -3

Alternate 2’s Complement Solution

- 2’s Complement:
  - 1011

    1. Flip Bits: 0100
    2. Add 1: 0101
    3. Interpret: -5

- 2’s Complement:
  - 1011

    -8 + 2 + 1 = -5

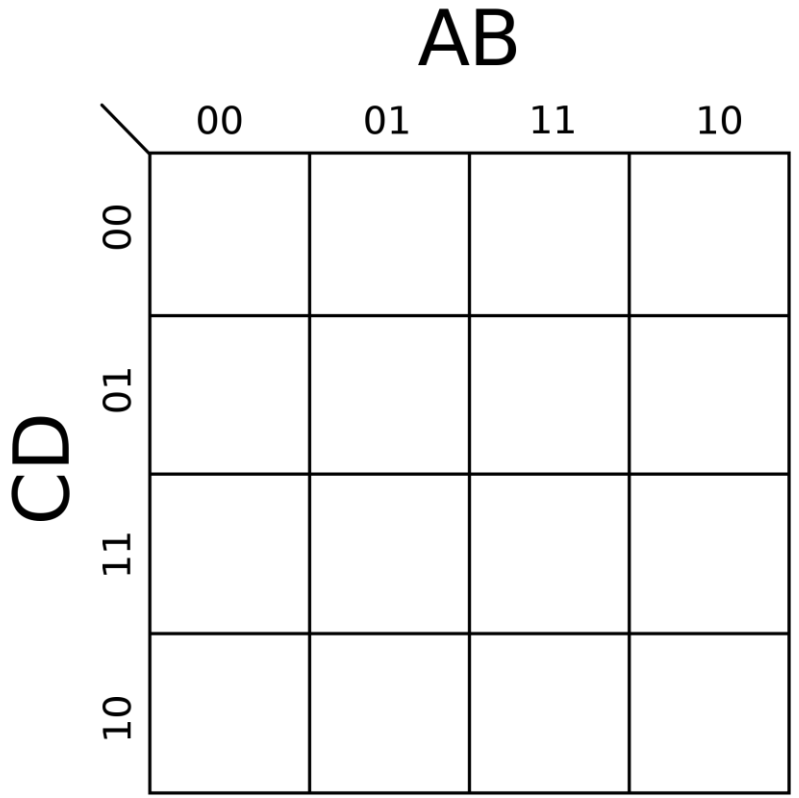
Arithmetic

- Think back to basic arithmetic in base 10

- Let’s just do some problems

SOP, POS, and K-Maps

- SOP
  - XY + YZ
- POS
  - (X+Y)(Y+Z)
- K-Map
  - Truth table to K-Map form
  - Equation to K-Map



Decoder

- Maps binary input to decimal output

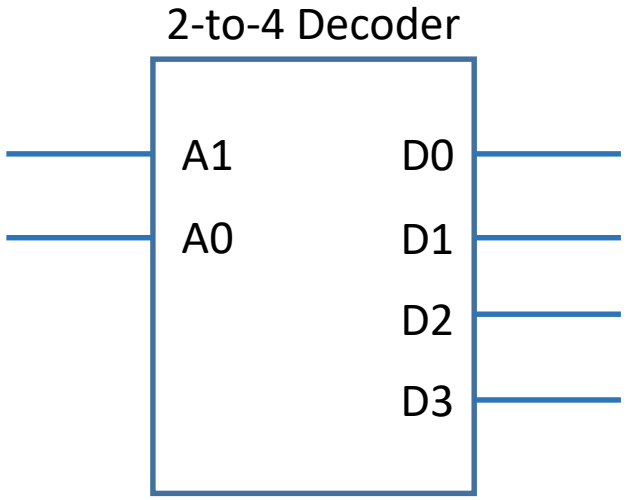
/A1 * /A0 → D0

/A1 * A0 → D1

A1 * /A0 → D2

A1 * A0 → D3

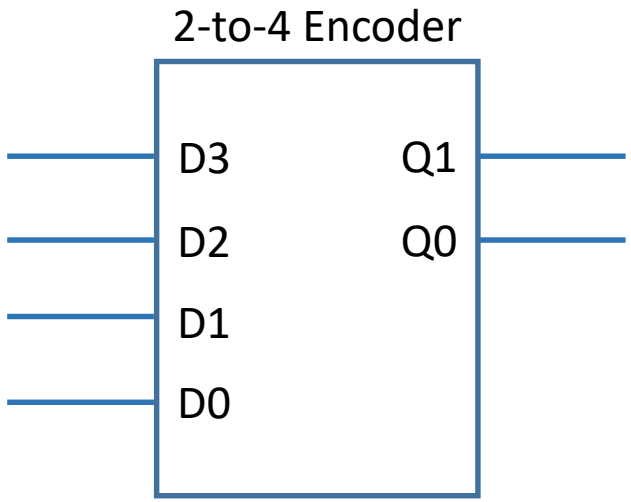
INPUTS		OUTPUTS			
A1	**A0**	**D3**	**D2**	**D1**	**D0**
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



Encoder

- Maps decimal input to binary output

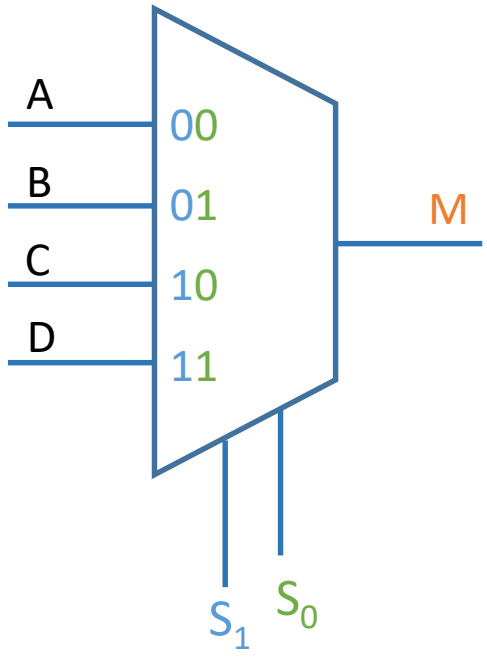
INPUTS				OUTPUTS	
D3	**D2**	**D1**	**D0**	**Q1**	**Q0**
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



Mux

- Multiple input, single output selection device

  - M = /S1 and /S0 and **A**
    + /S1 and S0 and **B**
    + S1 and /S0 and **C**
    + S1 and S0 and **D**

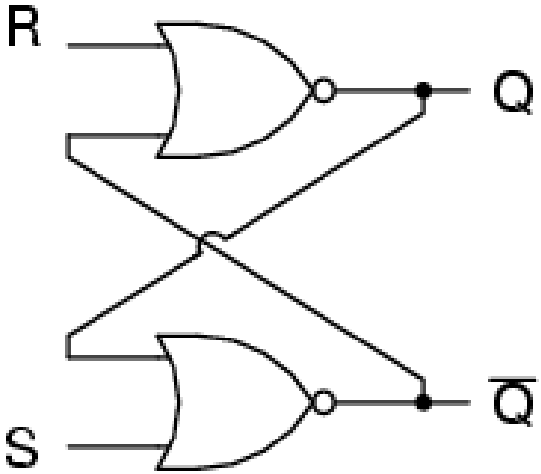


We can implement logic equations using multiplexers - example

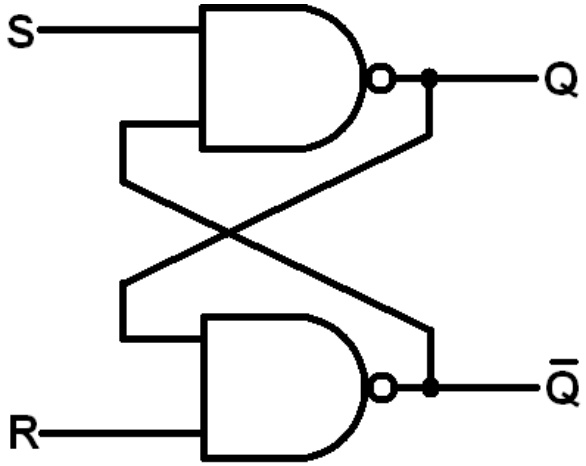
SR Latch

- NOR and NAND configurations

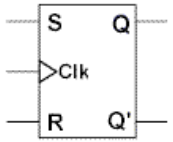
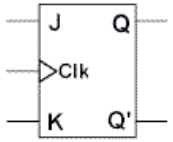
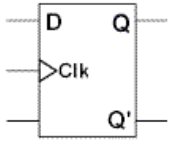
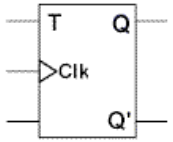
$Q^+ = S + /RQ$



S	R	Q	$\overline{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	UND	UND



Flip-Flops

FLIP-FLOP NAME	FLIP-FLOP SYMBOL	CHARACTERISTIC TABLE	CHARACTERISTIC EQUATION	EXCITATION TABLE
SR		S R Q(next) 0 0 Q 0 1 0 Reset 1 0 1 Set 1 1 ?	$Q_{(next)} = S + R'Q$ $SR = 0$	Q Q(next) S R 0 0 0 X 0 1 1 0 1 0 0 1 1 1 X 0
JK		J K Q(next) 0 0 Q Hold 0 1 0 Reset 1 0 1 Set 1 1 Q' Toggle	$Q_{(next)} = JQ' + K'Q$	Q Q(next) J K 0 0 0 X 0 1 1 X 1 0 X 1 1 1 X 0
D		D Q(next) 0 0 1 1	$Q_{(next)} = D$	Q Q(next) D 0 0 0 0 1 1 1 0 0 1 1 1
T		T Q(next) 0 Q 1 Q'	$Q_{(next)} = TQ' + T'Q$	Q Q(next) T 0 0 0 0 1 1 1 0 1 1 1 0

Delay

Toggle

Design a Counter Example

- **Example**: Count 7, 0, 3 using J-K Flip Flop and T-FF

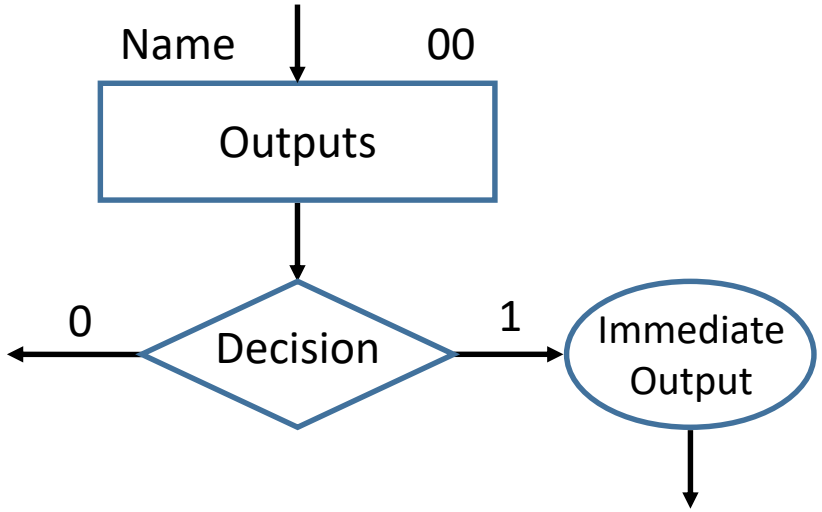
State Machines

- Moore Machines
  - Output only depends on current state
- Mealy Machine
  - Output depends on current state and input
  - Asynchronous
- **Example**: Detect sequence 010*1

ASM Diagrams

- Rectangle:
  - State box
  - Outputs go inside
- Diamond
  - Decision box, follows state box
  - Conditional Branch
  - Inputs go inside
- Oval:
  - Conditional (Mealy) outputs
  - Outputs go inside, asynchronous

- **Examples**:
  - Detect sequence 010*1
  - Washing Machine



Registers

- Just a collection of Flip-Flops

- **Example**:
  - Design 4-bit shift register with the following specs
    - Global enable
    - Asynchronous reset to 1010
    - We can give 4-bit input
    - Shift left and wrap
    - Shift right and sign extend