DIGITAL ELECTRONICS SYSTEM DESIGN

BROWN
School of Engineering

FALL 2019

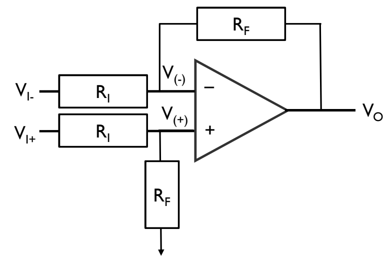
PROF. IRIS BAHAR (GIVEN BY JIWON CHOE)

NOVEMBER 6, 2019

LECTURE 17: BINARY ADDITION

SUMMING AMPLIFIER

- Output voltage follows the sum of two input voltages, one taken with the opposite sign



$$V_{(-)} \cong V_{(+)} = \frac{R_F}{R_I + R_F} V_{I+}$$

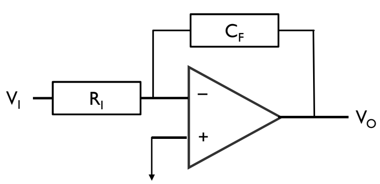
$$\frac{V_{I-} - V_{(-)}}{R_I} = \frac{V_{(-)} - V_o}{R_F}$$

$$V_o = ?$$

2

INTEGRATOR

- Output voltage is the time integral of the input voltage, with the opposite sign, and with a scale factor



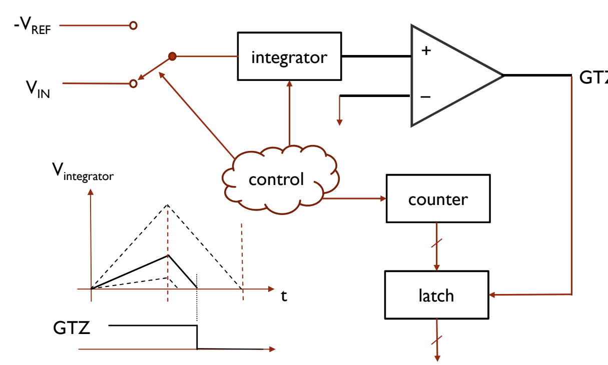
$$V_{(-)} \cong V_{(+)} = 0$$

$$\frac{V_I}{R_I} = C_F \frac{d}{dt}(0 - V_o)$$

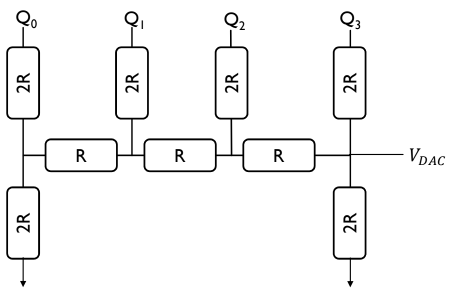
$$V_o = -\left(\frac{1}{R_I C_F}\right) \int V_I dt$$

3

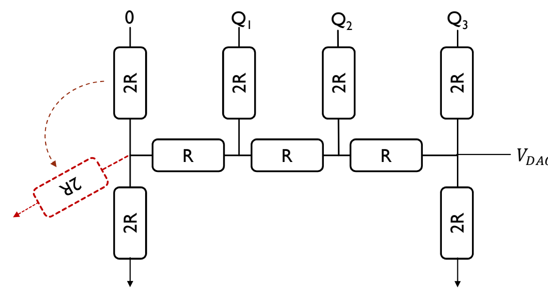
DUAL-SLOPE A/D CONVERTER



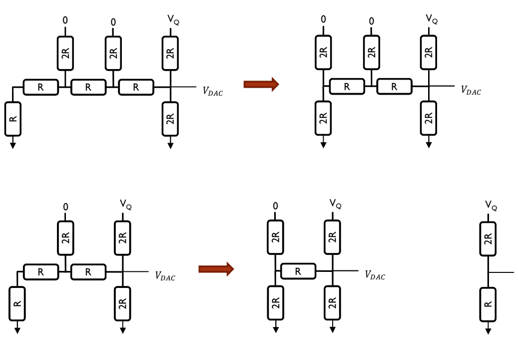
R-2R LADDER D/A CONVERTER



R-2R LADDER D/A CONVERTER

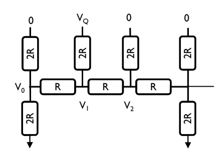


R-2R LADDER ANALYSIS BY SUPERPOSITION AND SERIES PARALLEL REDUCTION



$$V_{DAC} = \frac{R}{R + 2R} V_Q$$

R-2R LADDER ANALYSIS BY SUPERPOSITION AND SERIES PARALLEL REDUCTION



$V_{DAC} = ?$ → ? → ?

? → ? → ?

BINARY ADDITION

UNSIGNED BINARY NUMBERS

- For the binary number $b_{n-1}b_{n-2}\ldots b_1b_0.b_{-1}b_{-2}\ldots b_{-m}$ the decimal number is:

$$D = \sum_{i=-m}^{n-1} b_i 2^i$$

- Example:

  $101.001_2 = ?$

  $5 + 2^{-3} = 5.125$

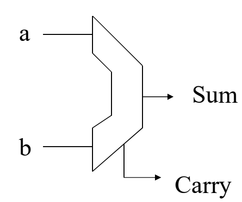
BINARY ADDITION

- Addition is an essential operation for all kinds of computing
- We need to understand how to do this for binary numbers
- We need to understand how to do this for positive and negative numbers
- We need to understand how to implement this efficiently in hardware

```
      5
  +   7
  -----
    1 2
Carry     Sum
```

```
  1   1   1   ← Carry bits
      1   0   1      5
  +   1   1   1      7
  -------------
  1   1   0   0     12
Carryout      Sums
```

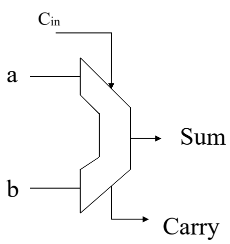
HALF ADDER



Truth Table

a	b	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

FULL ADDER

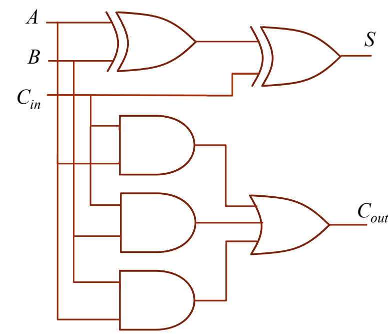


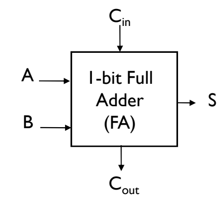
Truth Table

Id	a	b	c_{in}	carry	sum
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

- How do you express sum and carry as Boolean functions?

THE FULL ADDER



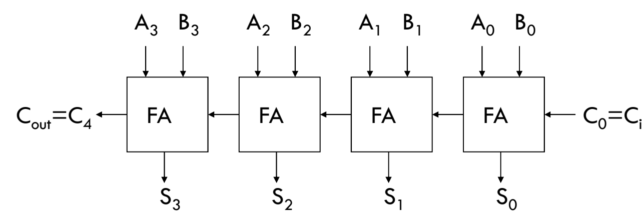


BUILDING A BINARY ADDER

```
  1  1  1  ← Carry bits
     1  0  1      5  (A)
 +   1  1  1      7  (B)
 ------------
  1  1  0  0     12  (S)
Carryout    Sums
```

- Inputs & outputs for the i^{th} bit position
  - Inputs: A_i, B_i, and C_i (carry-in)
  - Outputs: S_i (sum) and C_{i+1} (carry out)
- Carry out of a bit position is the carry in for the next bit position

THE RIPPLE CARRY ADDER



- The carry out of one stage ripples to the carry in of the next

WHAT ABOUT NEGATIVE NUMBERS?

- So far we have just considered unsigned numbers when converting from base 10 to binary.
- What about negative numbers and how do we add two signed numbers in binary?
- 3 ways of representing signed numbers:
  - Signed magnitude
  - 1's complement
  - 2's complement

SIGNED MAGNITUDE

- The Most Significant Bit (MSB) is the sign bit: 0 → positive, 1 → negative
- The rest of the bits define the magnitude
- Need to know how many bits are available to represent a number!
- Example: $(2)_{10} = (0010)_2 = (0\ 010)_{S\&M}$
  $(-2)_{10} = \qquad (1\ 010)_{S\&M}$
- Makes adding and subtracting a pain
  - Can't just add them regularly
- Also, two representations for zero (+0 and -0)

SIGNED MAGNITUDE ADDITION

```
   (1) 10    →    0001
 + (5) 10    →  + 0101
 ----------     ---------
   (6) 10         0110   (both positive, so a positive result)

   (-2) 10   →    1010
 + (-4) 10   →  + 1100
 ----------     ---------
   (-6) 10        1110   (both negative, so keep the negative sign)

   ( 4) 10   →    0100   (larger number – smaller number)
 + (-3) 10   →  + 1011   (keep the sign of the larger number)
 ----------     ---------
   ( 1) 10        0001   (signs are different → subtract smaller from
                           larger number, keep sign of larger number)
```

- Need a comparator to supplement adder/subtractor

1'S COMPLEMENT

- To negate a number, complement (invert, flip) each bit
- Example: $(4)_{10} = (0100)_2 = (0100)_{1\text{'s comp}}$
  $(-4)_{10} \qquad = (1011)_{1\text{'s comp}}$
- Like sign and magnitude, the high bit indicates the sign of the number
- What about adding and subtracting?

1'S COMPLEMENT ADD/SUBTRACT

```
  (-2) 10   →     1101
+ (-4) 10   →   + 1011
----------      ---------
  (-6) 10        11000  -- not right, (-6)10 = (1001)1's comp
                +    1  → add Cout back to LSB
                 ---------
                  1001  -- now it works

   (4) 10   →     0100
+ (-3) 10   →   + 1100
----------      ---------
   (1) 10        10000  -- not right, add Cout back to LSB
                +    1
                 ---------
                  0001  -- now it works
```

- Better than sign and magnitude (can subtract by adding the negative)
- But requires 2 addition operations (need to conditionally add C_{out})