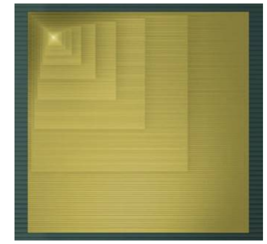
ECE 274 Digital Logic

Datapath Components – Subtractors, Two's Complement, Overflow, ALUs, Register Files

Digital Design 4.8 – 4.10

Digital Design

Chapter 4: Datapath Components

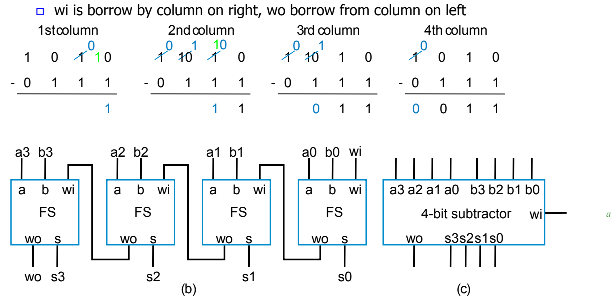


Slides to accompany the textbook *Digital Design*, First Edition, by Frank Vahid, John Wiley and Sons Publishers, 2007.
http://www.ddvahid.com

Datapath Components

Subtractor

- Can build subtractor as we built carry-ripple adder
  - Mimic subtraction by hand
  - Compute borrows from columns on left
    - Use full-subtractor component:
      - wi is borrow by column on right, wo borrow from column on left



Datapath Components

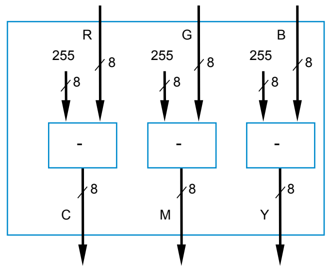
Subtractor Example: Color Space Converter – RGB to CMY

- Color
  - Often represented as weights of three colors: red, green, and blue (RGB)
    - Perhaps 8 bits each, so specific color is 24 bits
      - White: R=11111111, G=11111111, B=11111111
      - Black: R=00000000, G=00000000, B=00000000
      - Other colors: values in between, e.g., R=00111111, G=00000000, B=00001111 would be a reddish purple
  - Good for computer monitors, which mix red, green, and blue lights to form all colors

- Printers use opposite color scheme
  - Because inks *absorb* light
  - Use complementary colors of RGB: Cyan (absorbs red), reflects green and blue, Magenta (absorbs green), and Yellow (absorbs blue)

Datapath Components
Subtractor Example: Color Space Converter – RGB to CMY

- Printers must quickly convert RGB to CMY
  - C=255-R, M=255-G, Y=255-B
  - Use subtractors as shown

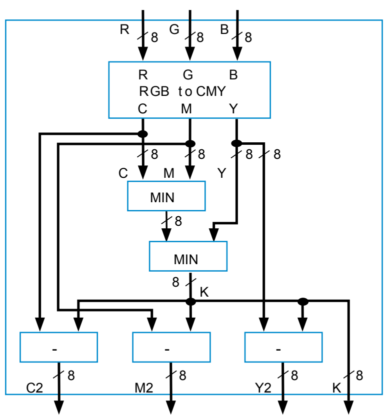


Datapath Components
Subtractor Example: Color Space Converter – RGB to CMYK

- Try to save colored inks
  - Expensive
  - Imperfect – mixing C, M, Y doesn't yield good-looking black
- Solution: Factor out the black or gray from the color, print that part using black ink
  - e.g., CMY of (250,200,200)= (200,200,200) + (50,0,0).
    - (200,200,200) is a dark gray – use black ink

Datapath Components
Subtractor Example: Color Space Converter – RGB to CMYK

- Call black part K
  - (200,200,200): K=200
  - (Letter "B" already used for blue)
- Compute minimum of C, M, Y values
  - Use MIN component designed earlier, using comparator and mux, to compute K
  - Output resulting K value, and subtract K value from C, M, and Y values
  - Ex: Input of (250,200,200) yields output of (50,0,0,200)



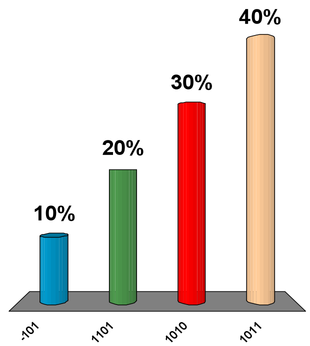
Representing Negative Numbers: Two's Complement

- Negative numbers common
  - How represent in binary?
- Signed-magnitude
  - Use leftmost bit for sign bit
    - So -5 would be:
      - 1101 using four bits
      - 10000101 using eight bits
- Better way: Two's complement
  - Big advantage: Allows us to perform subtraction using addition
  - Thus, only need adder component, no need for separate subtractor component!

Datapath Components
Two's Complement

- What is the 4-bit binary two's complement representation for the decimal number -5?

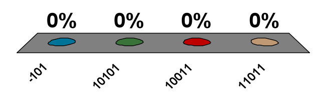
1. -0101
2. 1101
3. 1010
4. 1011



Datapath Components
Two's Complement

- What is the 5-bit binary two's complement representation for the decimal number -5?

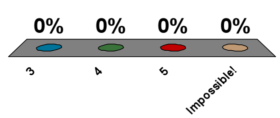
1. -00101
2. 10101
3. 10011
4. 11011



Datapath Components
Two's Complement

- How many bits are needed to represent the number -12 in binary?

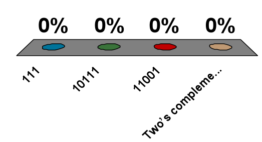
1. 3
2. 4
3. 5
4. Impossible!



Datapath Components
Two's Complement

- What is the 5-bit binary two's complement representation for the decimal number 7?

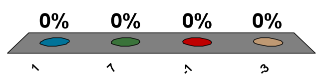
1. 00111
2. 10111
3. 11001
4. Two's complement can only represent negative numbers



Datapath Components
Two's Complement

- What is the decimal equivalent the two's complement binary number 111?

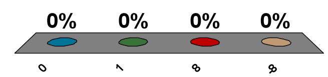
1. 1
2. 7
3. -1
4. -3



Datapath Components
Two's Complement

- What is the decimal equivalent the two's complement binary number 1000?

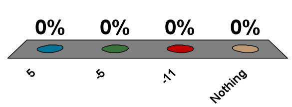
1. 0
2. 1
3. 8
4. -8



Datapath Components
Two's Complement

- What is the decimal equivalent the two's complement binary number 0101?

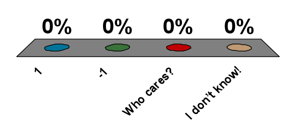
1. 5
2. -5
3. -11
4. Nothing



Datapath Components
Two's Complement

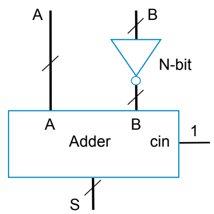
- What is the decimal equivalent the two's complement binary number 111111111111?

1. 1
2. -1
3. Who cares?
4. I don't know!



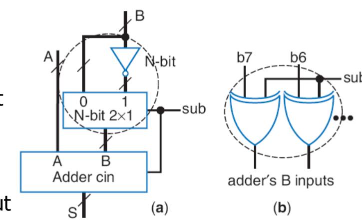
Datapath Components
Two's Complement Subtractor Built with an Adder

- Using two's complement
  A – B = A + (-B)
  = A + (two's complement of B)
  = A + invert_bits(B) + 1
- So build subtractor using adder by inverting B's bits, and setting carry in to 1



Datapath Components
Adder/Subtractor

- Adder/subtractor: control input determines whether add or subtract
  - Can use 2x1 mux – sub input passes either B or inverted B
  - Alternatively, can use XOR gates – if sub input is 0, B's bits pass through; if sub input is 1, XORs invert B's bits

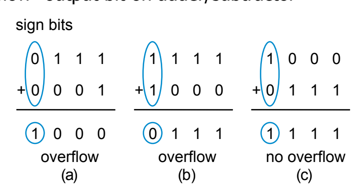


Datapath Components
Overflow

- Sometimes result can't be represented with given number of bits
  - Either too large magnitude of positive or negative
  - e.g., 4-bit two's complement addition of 0111+0001 (7+1=8). But 4-bit two's complement can't represent number >7
    - 0111+0001 = 1000 WRONG answer, 1000 in two's complement is -8, not +8
  - Adder/subtractor should indicate when overflow has occurred, so result can be discarded

Datapath Components
Overflow: Detecting Overflow: Method 1

- Assuming 4-bit two's complement numbers, can detect overflow by detecting when the two numbers' sign bits are the same but are different from the result's sign bit
  - If the two numbers' sign bits are different, overflow is impossible
    - Adding a positive and negative can't exceed largest magnitude positive or negative
- Simple circuit
  - overflow = a3'b3's3 + a3b3s3'
  - Include "overflow" output bit on adder/subtractor



If the numbers' sign bits have the same value, which differs from the result's sign bit, overflow has occurred.

Datapath Components
Overflow: Detecting Overflow: Method 2

- Even simpler method: Detect difference between carry-in to sign bit and carry-out from sign bit
- Yields simpler circuit: overflow = c3 xor c4

(a)	(b)	(c)
1 1 1	0 0 0	0 0 0
0 1 1 1	1 1 1 1	1 0 0 0
+ 0 0 0 1	+ 1 0 0 0	+ 0 1 1 1
0 1 0 0 0	1 0 1 1 1	0 1 1 1 1
overflow	overflow	no overflow

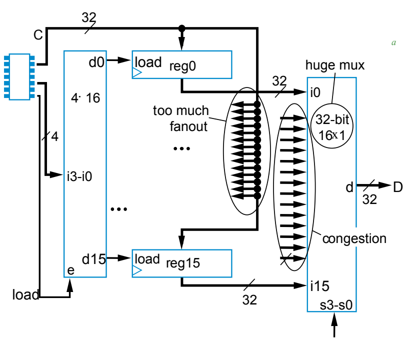
If the carry into the sign bit column differs from the carry out of that column, overflow has occurred.

Datapath Components
Magnitude Comparator Example: Minimum of Two Numbers

- Design a combinational component that computes the minimum of two 8-bit signed numbers

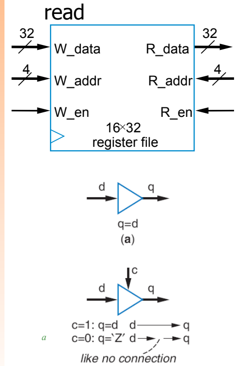
Datapath Components
Register Files

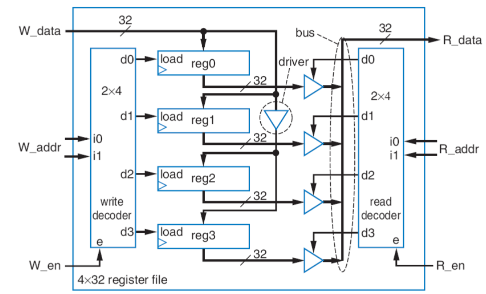
- ***MxN Register File***
  - Provides efficient access to M N-bit-wide registers
  - If we have many registers but only need access one or two at a time, a register file is more efficient
  - Ex: Above-mirror display (earlier example), but this time having 16 32-bit registers
    - Too many wires, and big mux is too slow



Datapath Components
Register Files

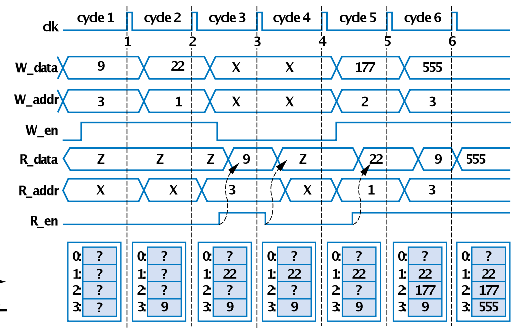
- Instead, want component that has one data input and one data output, and allows us to specify which internal register to write and which to read

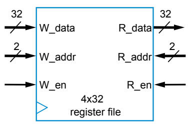




Datapath Components
Register Files: Timing Diagram

- Can write one register and read one register each clock cycle
  - May be same register





25

Datapath Components
Arithmetic-Logic Unit (ALU)

- ***ALU*** *(A brief overview)*
  - Component that can perform any of various arithmetic (add, subtract, increment, etc.) and logic (AND, OR, etc.) operations, based on control inputs
- Motivation:
  - Suppose want multi-function calculator that not only adds and subtracts, but also increments, ANDs, ORs, XORs, etc.

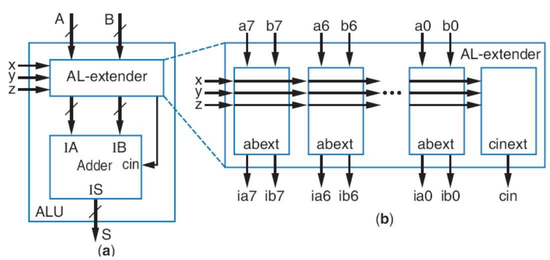
TABLE 4.2 Desired calculator operations

Inputs x	y	z	Operation	Sample output if A=00001111, B=00000101
0	0	0	S = A + B	S=00010100
0	0	1	S = A - B	S=00001010
0	1	0	S = A + 1	S=00010000
0	1	1	S = A	S=00001111
1	0	0	S = A AND B (bitwise AND)	S=00000101
1	0	1	S = A OR B (bitwise OR)	S=00001111
1	1	0	S = A XOR B (bitwise XOR)	S=00001010
1	1	1	S = NOT A (bitwise complement)	S=11110000

26

Datapath Components
Arithmetic-Logic Unit (ALU)

- More efficient design uses ALU
  - ALU design not just separate components multiplexed (same problem as previous slide!),
  - Instead, ALU design uses single adder, plus logic in front of adder's A and B inputs
    - Logic in front is called an arithmetic-logic extender
  - Extender modifies the A and B inputs such that desired operation will appear at output of the adder

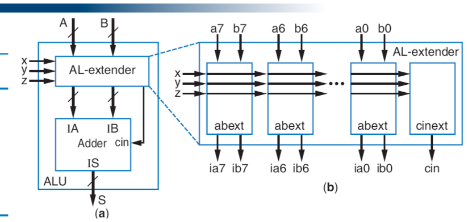


27

Datapath Components
Arithmetic-Logic Extender in Front of ALU

TABLE 4.2 Desired calculator operations

Inputs x	y	z	Operation	Sample output if A=00001111, B=00000101
0	0	0	S = A + B	S=00010100
0	0	1	S = A - B	S=00001010
0	1	0	S = A + 1	S=00010000
0	1	1	S = A	S=00001111
1	0	0	S = A AND B (bitwise AND)	S=00000101
1	0	1	S = A OR B (bitwise OR)	S=00001111
1	1	0	S = A XOR B (bitwise XOR)	S=00001010
1	1	1	S = NOT A (bitwise complement)	S=11110000



- xyz=000: Want S=A+B – just pass a to ia, b to ib, and set cin=0
- xyz=001: Want S=A-B – pass a to ia, b' to ib, and set cin=1
- xyz=010: Want S=A+1 – pass a to ia, set ib=0, and set cin=1
- xyz=011: Want S=A – pass a to ia, set ib=0, and set cin=0
- xyz=1000: Want S=A AND B – set ia=a*b, b=0, and cin=0
- others: likewise
- Based on above, create logic for ia(x,y,z,a,b) and ib(x,y,z,a,b) for each abext, and create logic for cin(x,y,z), to complete design of the AL-extender component

28

Datapath Components *(An RTL Preview!!)*
Counter Example: Timer

- A type of counter used to measure time
  - If we know the counter's clock frequency and the count, we know the time that's been counted
- Example: Compute car's speed using two sensors
  - First sensor (a) clears and starts timer
  - Second sensor (b) stops timer
  - Assuming clock of 1kHz, timer output represents time to travel between sensors. Knowing the distance, we can compute speed