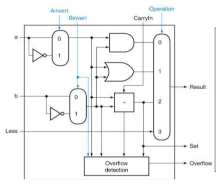
Lab 2 - More Practice for Logic Design

Tuesday, February 9, 2021 2:42 PM

JACOB WHITLOW

1. Overflow condition is checked in the 1-bit ALU for the most significant bit (sign bit).



Overflow cases:
1. adding 2 positive #s → sum is negative #;
2. adding 2 negative #s → sum is positive #;
(3=1. subtracting a neg# from a pos# → result neg#;)
(4=2. subtracting a pos# from a neg# → result pos#;)

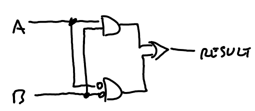
Overflow checking logic:
if (sign of op1 == sign of op2)
if (sign of op != sign of sum)
→ overflow

Draw a complete schematic diagram for the overflow detection component, using only three basic gates (2-input AND, 2-input OR, and negator).

SIGN BITS

A	B	RESULT
0	0	1
1	1	0

$(\bar{A}\cdot\bar{B}) + (A\cdot B)$



2. Prove that the following logic for checking overflow condition is equivalent to the logic shown in #1.

if ('carry_in to sign_bit' != 'carry_out from sign_bit') → overflow

A	B	CIN	RESULT (A+B+CIN) ← ADDITION	COUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

From this table we can select the occurrences where A == B != result (highlighted in pink), then we can select the occurrences where cin != cout (highlighted in blue). Once this is done it is apparent that these two logical conditions occur given the same inputs, meaning they are equivalent, therefore

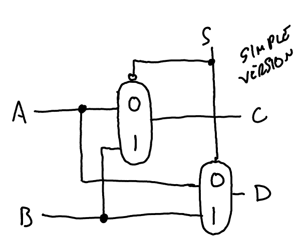
((cin to sign bit) != (cout from sign bit)) == ((sign op1 == sign op2) && (sign op != sign sum)) == overflow

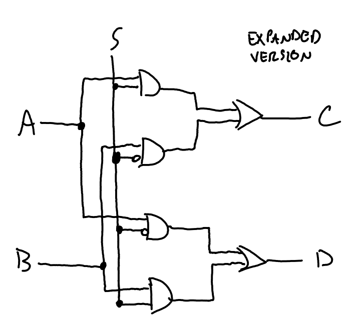
3. Implement (draw a schematic diagram for your design) a switching network that has two inputs (A and B), two outputs (C and D), and a control input (S).
The logic for the switching network is:

if (S == 1), the network is in the pass-through mode, i.e., A → C and B → D;
if (S == 0), the network is in the crossing mode, i.e., B → C and A → D;

Hint: use two 2x1 mux's;
Please show your schematic diagram using only three basic 2-input gates.

A	B	S	C	D
0	0	1	0	0
0	0	0	0	0
0	1	1	0	1
0	1	0	1	0
1	0	1	1	0
1	0	0	0	1
1	1	0	1	1
1	1	1	1	1





4. Write Amdahl's law (explain the terms used) and solve the following problem.

Suppose that we enhance a computer system to make all floating-point instructions run faster than the original version. Assume that a benchmark program consists of floating point instructions (25%) and other instructions (75%).
To achieve the speedup of 1.7 for running this benchmark program, what should be the speedup of the enhanced mode (floating point part)?

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}}$$

SPEEDUP OF SYSTEM

FRACTION OF ENHANCED MODE

SPEEDUP OF ENHANCED MODE

$$1.7 = \frac{1}{(1-\frac{1}{4}) + \frac{1/4}{n}}$$

$$\frac{1}{1.7} = 0.75 + \frac{0.25}{n}$$

$$-0.162 = \frac{0.25}{n}$$

$$n = -1.545$$

This negative speedup is impossible and means that given the current conditions there is no way to achieve a speedup of the system of 1.7 by only increasing the speed of 25% of the performed operations.