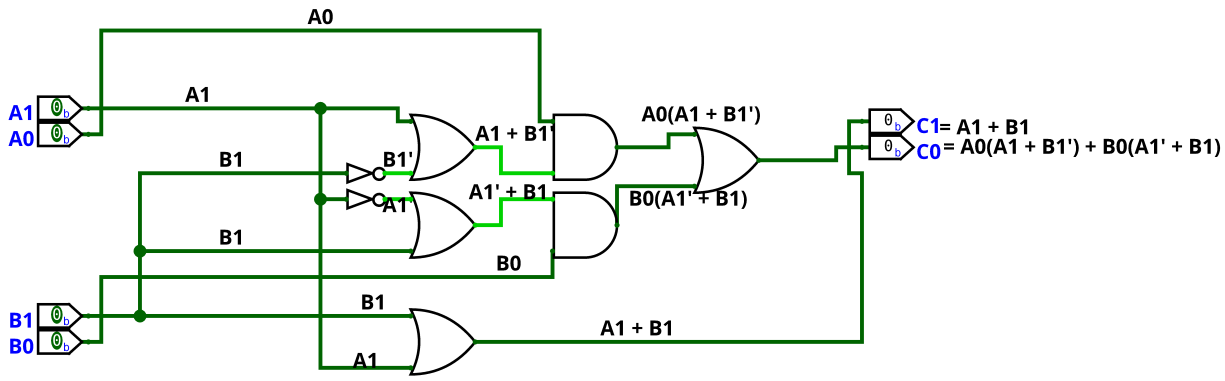
Lab 4 - Combinational Iterative Design

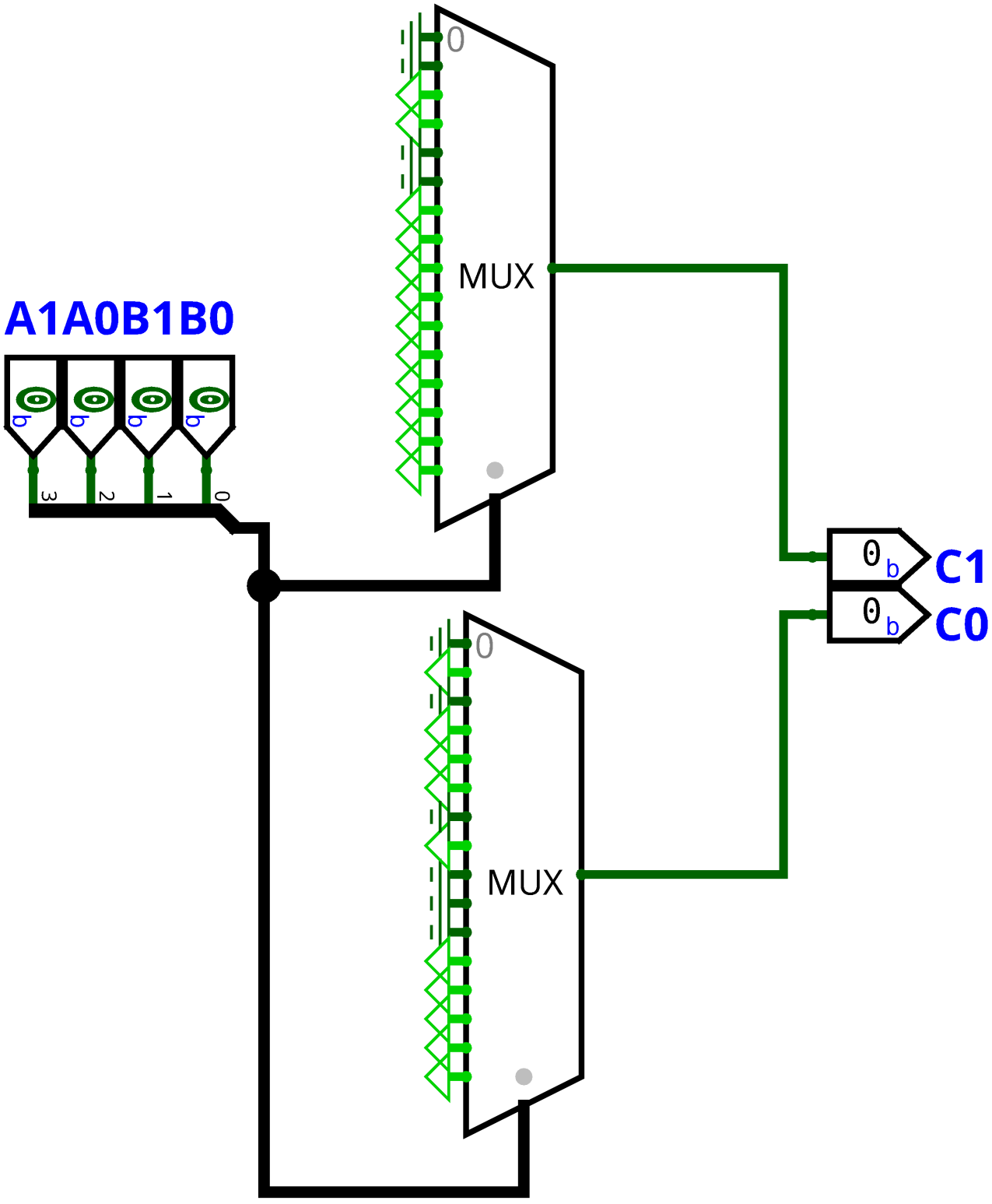
- Objectives
  - Implement several different devices using a general iterative approach
    - Design #0 - A 2-bit maximum value selector using AND/OR/NOT logic devices
    - Design #1 - A 2-bit maximum value selector using two 1-bit 16-to-1 multiplexers
    - Design #2 - A 2-bit maximum value selector that uses only 1-bit 2-to-1 multiplexers
    - Design #3 - A 2-bit maximum value selector that uses only NAND gates
    - Design #4 - A 1-bit maximum value selector to be used in larger circuit problems using iterative design
    - Design #5 - Construct a 4-bit maximum value selector given two 4-bit inputs using the 1-bit maximum value constructor from design #4
- Design #0
  - Reused circuit from Lab 3
  - Figure #1: Circuit Design #0



- Design #1
  - Realize a solution for a 2-bit maximum value selector that uses two 1-bit 16-to-1 multiplexers
    - One MUX for C1, one for C0
    - Use no other discrete logic devices
  - Design
    - Used two 1-bit 16-to-1 MUX devices, one for the C1 output, one for the C0 output
    - A1A0B1B0 input pins are connected to a splitter
      - The wire output is split and fed to each MUX devices' select terminal
      - The corresponding truth table values for C1/C0 are mapped to the MUX device with either a ground pin or a power pin
  - Table #1: Truth Table from Lab 3, applied to design #1 circuit

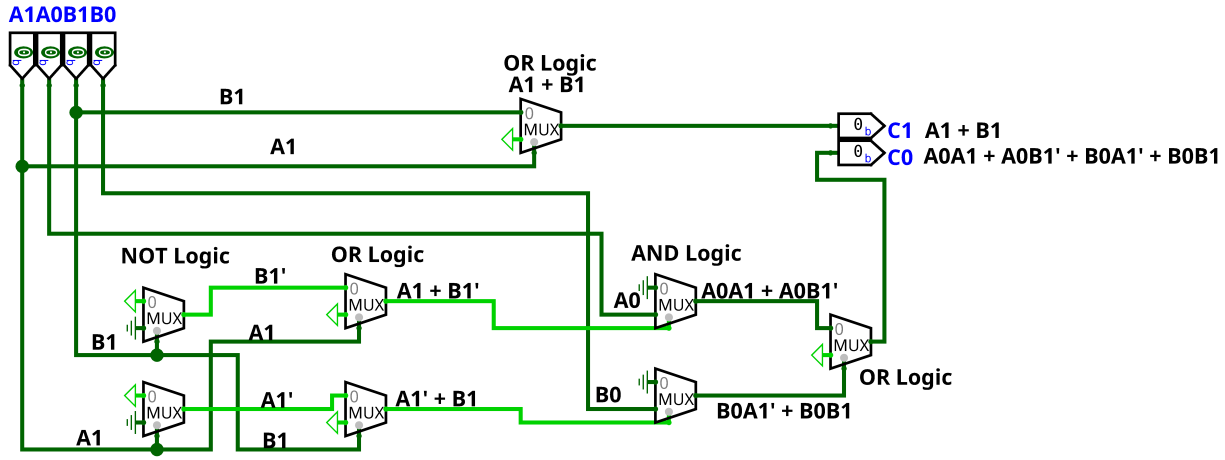
A1	A0	B1	B0	C1	C0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

- Figure #2: Circuit for Design #1

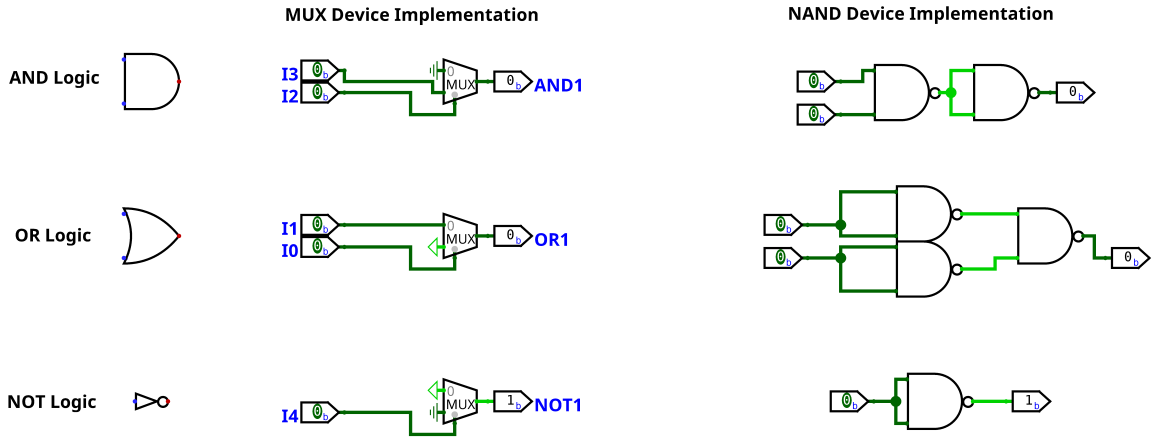


- Design #2
  - Realize a solution for a 2-bit maximum value selector that uses only 1-bit 2-to-1 multiplexers
    - Use no other logic device
  - Design:

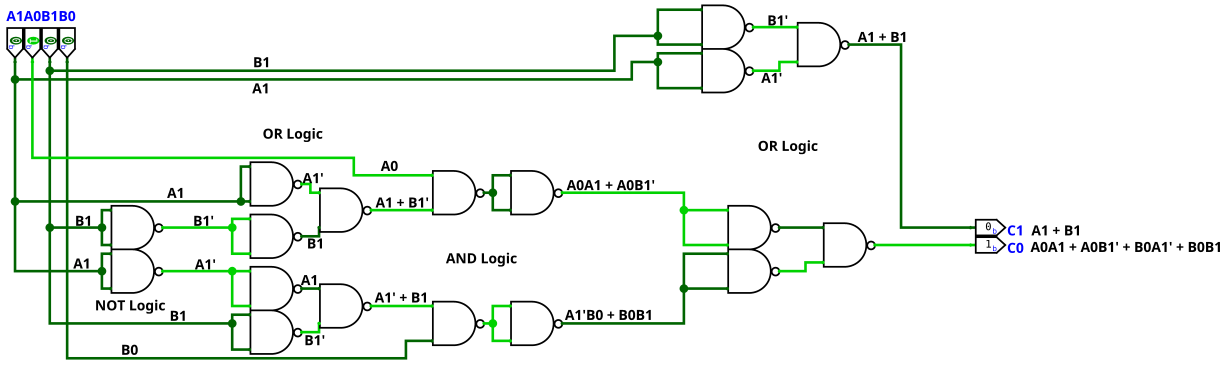
- Using the previously determined Boolean expressions from lab 3 for a circuit with the same functions:
  - C1 = A1 + B1
  - C0 = A0(A1 + B1') + B0(A1' + B1)
- And the logical functions that can be implemented with MUX devices
  - OR
    - Input pin 1 terminated to the MUX *Input terminal 0*
    - Input pin 2 terminated to the MUX *select* terminal
    - Supplying constant power to the MUX *Input terminal 1*
      - The output of the MUX device will be 1 if one or both input devices are 1
  - AND
    - Input pin 1 terminated to the MUX *Input terminal 1*
    - Input pin 2 terminated to the MUX *select* terminal
    - Ground the MUX *Input terminal 0*
      - The output of the MUX device will be 1 only when both inputs are 1
  - NOT
    - The MUX *terminal 0* is supplied constant power
    - The MUX *terminal 1* is grounded
    - The Input pin is terminated to the MUX *select* terminal
      - The output of the MUX device is 1 when the input device is 0. The output of the MUX device is 0 when the input device is 1
- Implementation:
  - I was able to implement this circuit by following the Boolean expressions and implementing the logical operators using the MUX devices wired as mentioned above
- Figure #3: Design 2 Circuit



- Design #3
  - Realize a new implementation for a 2-bit maximum value selector that uses only NAND gates
    - No other logic devices may be used
  - Design:
    - Using the previously determined Boolean expressions from lab 3 for a circuit with the same functions:
      - C1 = A1 + B1
      - C0 = A0(A1 + B1') + B0(A1' + B1)
    - In conjunction with the properties of NAND logic
    - Figure #4: Logical operation of NAND and MUX devices

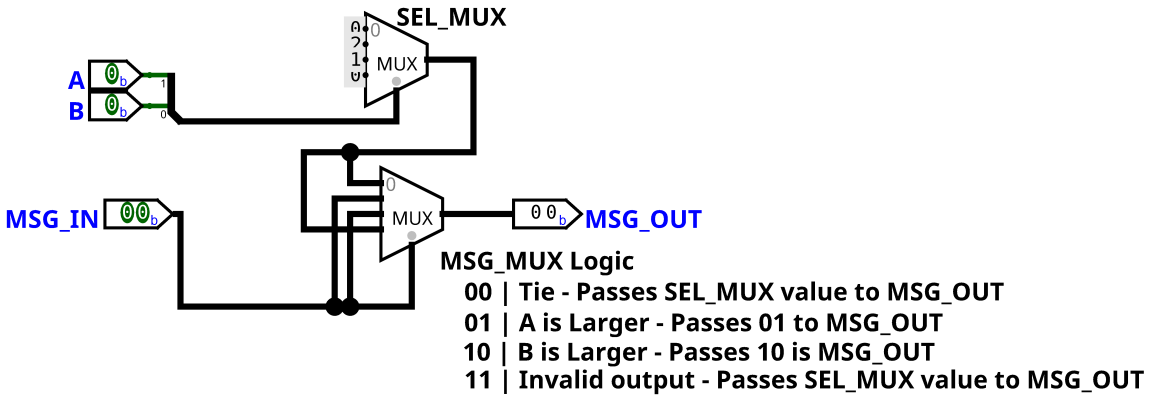


  - Implementation:
    - Similarly to design #2, I was able to follow the Boolean logic to build the circuit, replacing the AND/OR/NOT devices with the corresponding NAND device configuration
  - Figure #5: Design #3 Circuit



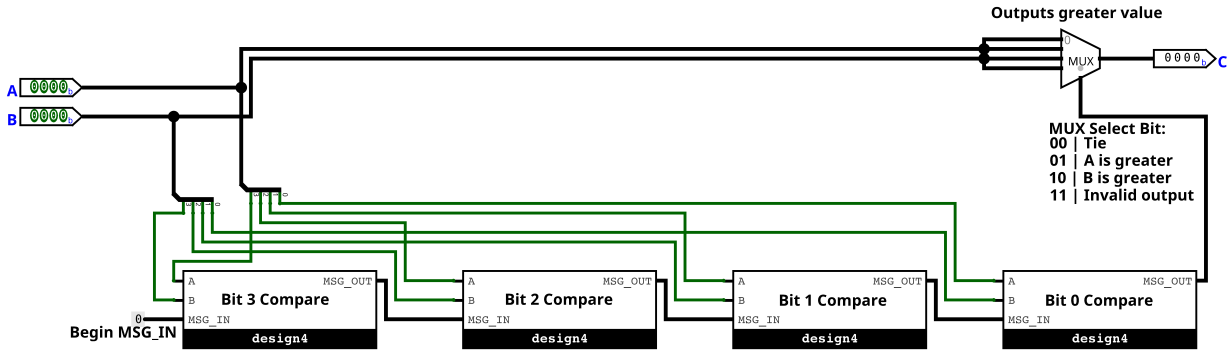
- Design #4
  - Design a 1-bit maximum value selector to be used in larger circuit problems using iterative design
    - May require additional input/output bits to communicate information between devices

- Design:
  - This device will consist of *two* MUX devices
    - One MUX device will function as our maximum value selector
    - The other MUX device will function as our communication device
  - Logic:
    - The SEL_MUX will be responsible for comparing the two 1-bit values
      - Both values will terminate into a splitter, which will control the *select* terminal of the SEL_MUX
      - SEL_MUX Output:
        - A = 0 | B = 0 | SEL_MUX Out = 00
        - A = 0 | B = 1 | SEL_MUX Out = 01
        - A = 1 | B = 0 | SEL_MUX Out = 10
        - A = 1 | B = 1 | SEL_MUX Out = 00
      - The MSG_MUX output is determined by the MSG_IN 2-bit select
        - If the MSG_IN has a value of 01 or 10, the MUX will pass those values forward to the MSG_OUT terminal
        - Otherwise if the MSG_IN has a value of 11 or 00, the MUX will output the value from its input 0, which is the SEL_MUX output
          - The MSG_OUT will mirror the SEL_MUX output value
  - Implementation:
    - The circuit was relatively easy to construct
      - For the SEL_MUX, I used constant 2-bit value for the input pins in order to communicate which of the two compared values are larger
  - Figure #6: Circuit Diagram for Design #4



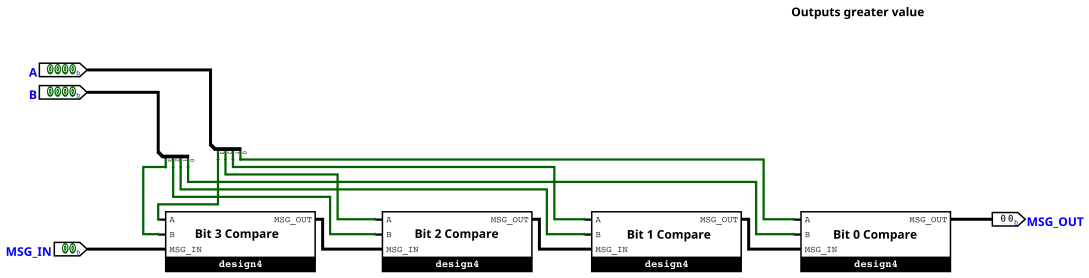
- Design #5
  - Construct a 4-bit maximum value selector given two 4-bit inputs using the 1-bit maximum value constructor from design #4
    - Inputs: A[3:0] & B[3:0]
    - Output: C[3:0]

- Design:
  - The circuit will consist of blocks, in this case 4, from the Design #4
    - The 4-bit inputs are to be split into single bit pairs, so that each individual bit is compared
    - The bits are to be evaluated from most significant bit to least significant bit
      - The logic behind this is due to the fact that if for example 1001 and 0101 are compared, evaluating from the LSB gives no indication of which bit string is larger until the final bit is compared
      - Comparing from MSB to LSB will allow for the quickest and most accurate evaluation of the values
    - Message bits are linked from one device to the next
      - The initial MSG_IN is a constant 00 value to "start" the comparator circuit
      - The final MSG_OUT at the end of the chain is terminated into a MUX devices which outputs the 4-bit value of A or B, dependent on which is greater
  - Figure #7: Circuit Diagram for Design #5



- Further exploration of a 32-bit maximum value selector
  - Given the problem of comparing two 32-bit values, the approach I would take to implement this would be:
    - Similar to Design's #4 & #5
      - Taking the circuit design #5 I would eliminate the function of outputting the greater of the compared values for the moment and retain the MSG_OUT outputting a value corresponding to which value, A or B, is larger
      - The new design would use a chain of these devices, 8 to be specific, with the same structure as the original Design #5
        - The circuit would only differ in structure from the original Design #5 for input/output pins which are now 32-bits. The splitters, would now split a 32-bit value into 8 4-bit values, which would be terminated to the corresponding devices for comparison
    - Implemented Design(not required but pretty easy to implement)

- Figure #8: Modified Design #5



- Figure #9: Implemented Final Circuit for 32-bit max value selector