Function Unit

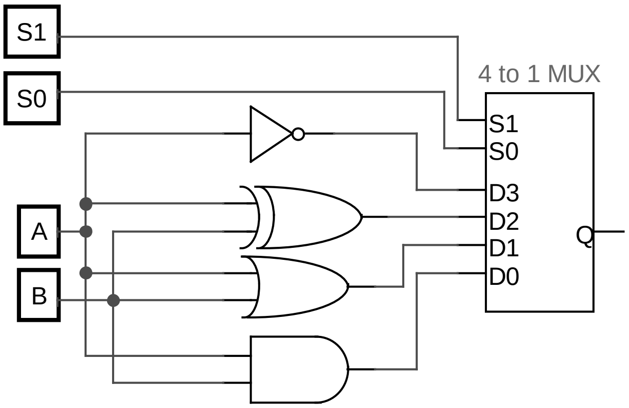
June 6

- The ALU & shifter
- The ALU: arithmetic, logic

Logic

- Does logic on its inputs, bit by bit
- It will have two select bits: S_1, S_0

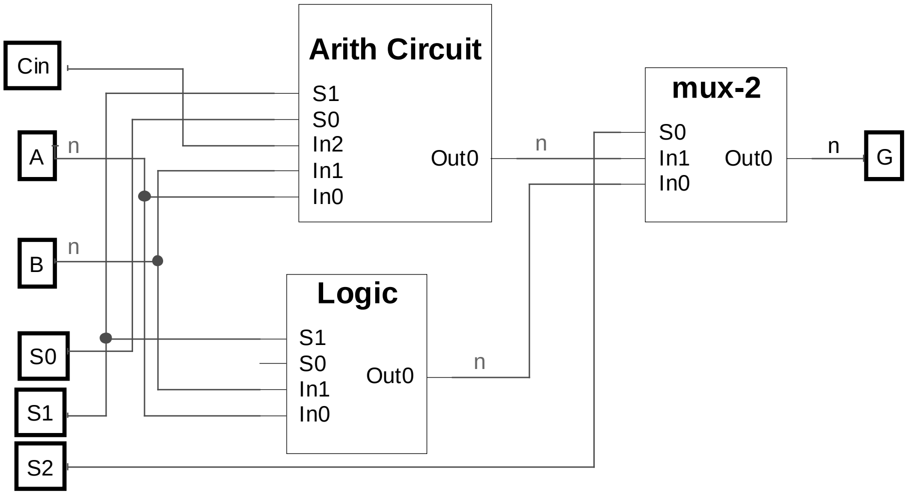
S_1	S_0	operation
0	0	AND
0	1	OR
1	0	XOR
1	1	NOT (A)



construction:

[n copies of this –one for each bit in A and B]

The ALU



The four selection inputs S_2, S_1, S_0 C_{in} control the operation

The Shifter

- Takes the B operand as input
- A doesn't enter the shifter

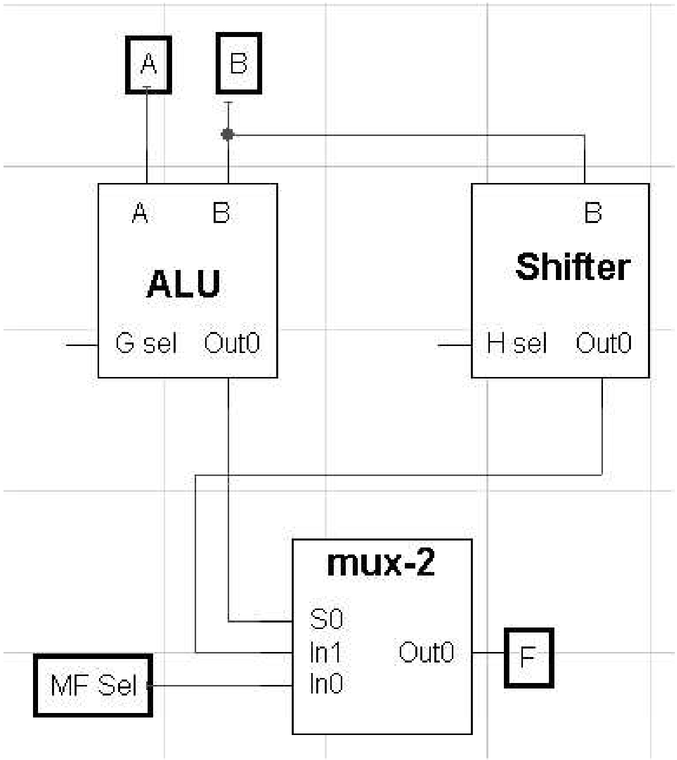
- Two control inputs S_1, S_0

S_1, S_0 Operations

S_1	S_0	operation
0	0	B
0	1	Shift right B
1	0	Shift left B
1	1	undefined

Implemented with multiplexers (fig 7-16)

The Function Unit



- MFsel: doing as ALU or shifter op.
- G sel: S_2, S_1, S_0 C_{in}
- H sel: S_1, S_0

We can combine these control signals into a 5 bit function select (FS)

  - First bit (FS (4)) is MFsel
  - If MFsel = 0, FS(3:0) is G sel
  - If MFsel = 1 FS (3:2) is H sel
    - [FS(1:0) unused]

e.g., fs = 00101

- MF sel = 0 ALU
- $S_2 = 0$ Arithmetic Circuit
- $S_1=1$ $S_0 = 0$, $C_{in} = 1$ → subtract
- E.g. FS = 10100

- MF sel = 1 shift
- Hsel = 01 shift right

(Last 00 ignored)

Status Bits

- The control unit needs some status signals, which indicate result of previous operation to implement instruction like BRZ (branch if zero)
- We can calculate these from the : function unit outputs
  - Z: results was zero
  - Diagram
  - Zero detects is really a nor gate:
  - C: carry
  - Take C_{out} from arithmetic circuit (or select between $C_{out)}$ and the shifters carry bit)
  - N: negative take the high bit from output
  - V: overflow in the adder: C_{out} XOR the last internal carry

Limitations

- has a fairly limited set of operations: + - inc, dec, sr, sl
- we can either recreate a very minimal machine language with these ops or
  - make a more complicated data math
  - do multiple cycle control
- it also has a long propagation delay