Subtractor

Half subtractor

* Is a comb logic ckt that subtracts 2 bits and produces their difference and borrow (B)

|  |  |
| --- | --- |
| X y | B D |
| 0 0 | 0 0 |
| 0 1 | 1 1 |
| 1 0 | 0 1 |
| 1 1 | 0 0 |

* D = X'Y + XY'
  + Or x (exclusive or) y
* B = X'Y

Full Subtractor

|  |  |
| --- | --- |
| X Y Z | B D |
| 0 0 0 | 0 0 |
| 0 0 1 | 1 1 |
| 0 1 0 | 1 1 |
| 0 1 1 | 1 1 |
| 1 0 0 | 0 1 |
| 1 0 1 | 0 0 |
| 1 1 0 | 0 0 |
| 1 1 1 | 1 1 |

* Kmap

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x\yz | 00 | 01 | 11 | 10 |
| 0 |  | 1 |  | 1 |
| 1 | 1 |  | 1 |  |

* D = X'Y'Z + X'YZ' + XY'Z' + XYZ
* KMAP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X\YZ | 00 | 01 | 11 | 10 |
| 0 |  | 1 | 1 | 1 |
| 1 |  |  | 1 |  |

* B = X'Z + X'Y + YZ

Binary Adder and Subtractor

* A – B = A + (-B) (<--- 2s compliment of B)
  + Take the 2s compl by taking the 1s compl using inverters and add 1 can be added by the input carry signal
  + She drew the circuit for the adder and subtractor on the board

Decoder

* IS a comb ckt that converts binary info from N\_input lines to a max 2^n output lines
* EX: 3x8 decoder
* (drew ckt on board)
  + Xyz -> D(1-8)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| xyz | D0 | d1 | d2 | d3 | d4 | d5 | d6 | d7 |
| 000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 010 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 011 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 100 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 101 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 110 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

* + D0 = X'Y'Z'
  + D1 = X'Y'Z
  + D2 = X'YZ'
  + D3 = X'YZ
  + D4 = XY'Z'
  + D5 = XY'Z
  + D6 = XYZ'
  + D7 = XYZ

Pr: Implemant a full adder ckt with a decoder

* S (x,y,z) = (sigma) (1,2,4,7)
* C (x,y,z) = (sigma) (3,5,6,7)
* X y z feed into the decoder, and the decoder has 8 outputs. And you or the outputs of (1,2,4,7) to get the sum and you or the outputs (3,5,6,7) to get the carry.

EX: build a 2x4 line decoder with an Enable bit and NAND gates

* When E = 1 all outputs are 1 and the ckt is disabled as a decoder

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| E | A | B | D0 | D1 | D2 | D3 |
| 1 | X | X | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |

Demultiplexer

* IS a ckt that recieves info on a single lime and transmits info on one of 2^n possible output lines
* E is input, D(0-3) outputs. And A B as select. On a 1x4 multiplexer ckt

Encoder

* Inverse operation of the decoder 2^n input lines -> n output lines. The output lines generate the binary code that corresponds to the input value
* EX: Encoder octal to binary

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | XYZ |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 000 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 001 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 010 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 011 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 100 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 101 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 110 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 111 |

* + X = D4 + D5 + D6 + D7
  + Y = D2 + D3 + D6 + D7
  + Z = D1 + D3 + D5 + D7
    - Established a higher priority for inputs with a higher subscript values.

Priority Encoder

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| D0 | D1 | D2 | D3 | X | Y | V(VALID BIT) |
| 0 | 0 | 0 | 0 | X | X | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| X | 1 | 0 | 0 | 0 | 1 | 1 |
| X | X | 1 | 0 | 1 | 0 | 1 |
| X | X | X | 1 | 1 | 1 | 1 |

* Kmap - X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D0,D1\D2,D3 | 00 | 01 | 11 | 10 |
| 00 | X | 1 | 1 | 1 |
| 01 |  | 1 | 1 | 1 |
| 11 |  | 1 | 1 | 1 |
| 10 |  | 1 | 1 | 1 |

* + X = D2 + D3
* Kmap – Y

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D0,D1\D2,D3 | 00 | 01 | 11 | 10 |
| 00 | X | 1 | 1 |  |
| 01 | 1 | 1 | 1 |  |
| 11 | 1 | 1 | 1 |  |
| 10 |  | 1 | 1 |  |

* + Y = D3 + D1D2'
* Kmap – V

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D0,D1\D2,D3 | 00 | 01 | 11 | 10 |
| 00 |  | 1 | 1 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

* + V = D2 + D3 + D1 + D0

Multiplexer (=Data Selector)

* Takes inputs and has selectors that select an input and has that be outputted

|  |  |  |
| --- | --- | --- |
| S1 | S0 | Y |
| 0 | 0 | I0 |
| 0 | 1 | I1 |
| 1 | 0 | I2 |
| 1 | 1 | I3 |

Boolean Function Implementation

* Implement Function with MUX that has N-1 selections inputs
  + First N-1 var of function are connected to selection inputs of MUX
  + Second the remaining single var is used for data Inputs
* Ex: F(x,y,z) = (sigma) (1,2,6,7)

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | Z | F |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

* + It’s a 4x1 multiplexer
  + X and y are the selectors
  + Yellow – F = z - I0
  + Green – F = z' - I1
  + Blue – F = 0 - I2
  + Red – F = 1 – I3
* EX: F(A,B,C,D) = (SIGMA) (1,3,4,11,12,13,14,15)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | F |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

* + It’s a 8x1 multiplexer
  + ABC are multiplexers
  + F is outputs
  + Dark red – F = D
  + Red – F = D
  + Orange – F = D'
  + Yellow – F = 0
  + Light green – F = 0
  + Green – F = D
  + Light blue – F = 1
  + Blue – F = 1