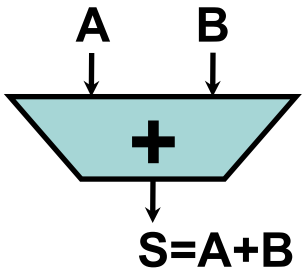
Combinational Circuits

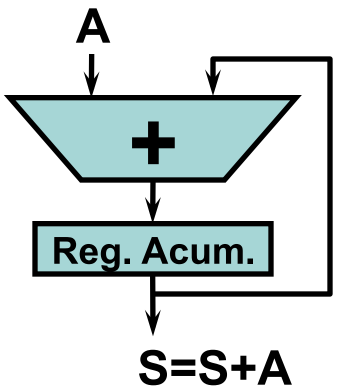
© Luis Entrena, Celia López,
Mario García, Enrique San Millán
Universidad Carlos III de Madrid

Combinational and sequential circuits

- Combinational:
  - Outputs affected only by inputs
  - Example: adder



- Sequential:
- Outputs affected by inputs and state
  - Example: adder-accumulator



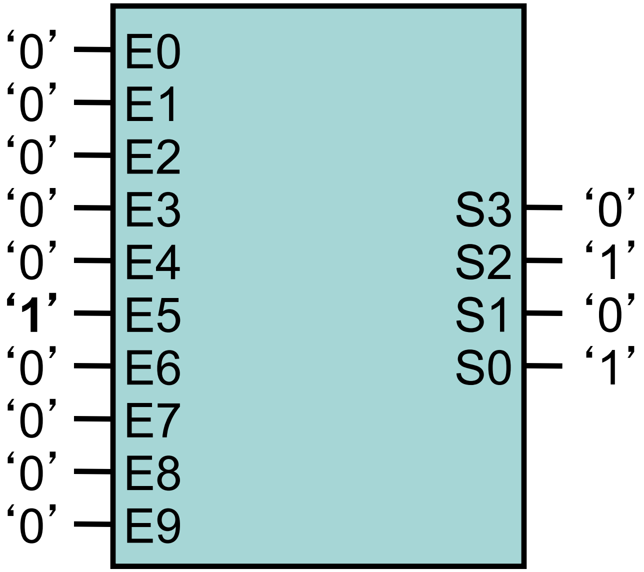
Outline

- Encoders
- Decoders
- Multiplexers
- Demultiplexers
- Comparators

- Functionality
- Implementation
- Asociation
- Function builders
- Utilities

1. Encoders

- Definition:
  - A combinational circuit allows to translate and active level of an input into an encoded value
- Example: keypad
  - Inputs: digits 0-9
  - Outputs: 4-bit binary code



E5 active => S="0101" (=5)

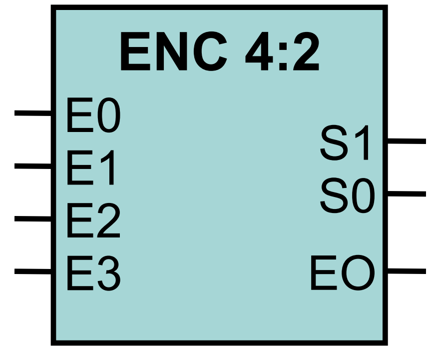
No priority encoders

- Restrictions
  - Only one input can be active
  - Several active inputs may lead to erroneus output
- Logic functions
  - $S_3=E_8+E_9$
  - $S_2=E_4+E_5+E_6+E_7$
  - $S_1=E_2+E_3+E_6+E_7$
  - $S_0=E_1+E_3+E_5+E_7+E_9$
- Problems:
  - E1 and E4 active lead to 5 code
  - No active inputs lead to 0 code, same as E_0 active

Active input	$S_3S_2S_1S_0$
E_0	0000
E_1	0001
E_2	0010
E_3	0011
E_4	0100
E_5	0101
E_6	0110
E_7	0111
E_8	1000
E_9	1001

Example: no priority 4:2 encoder

- M:N ⇒ 'M' inputs, 'N' outputs
- EO: "Enable Output"
  - Used to distinguish the no active inputs and E0 active input cases.
  - Used to chain several encoders
- Not considered cases
  - Any multiple activation
  - Don't care output values



E_3	E_2	E_1	E_0	S_1	S_0	EO
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
1	0	0	0	1	1	0
0	0	0	0	0	0	1
Others				X	X	X

Example:
no priority 4:2 encoder

E_3	E_2	E_1	E_0	S_1	S_0	EO
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
1	0	0	0	1	1	0
0	0	0	0	0	0	1
Others				X	X	X

E_1E_0 / E_3E_2	00	01	11	10
00	0	0	X	0
01	1	X	X	X
11	X	X	X	X
10	1	X	X	X

$$S_1 = E_2 + E_3$$

E_1E_0 / E_3E_2	00	01	11	10
00	1	0	X	0
01	0	X	X	X
11	X	X	X	X
10	0	X	X	X

$$EO = \overline{E_3}\,\overline{E_2}\,\overline{E_1}\,\overline{E_0}$$

E_1E_0 / E_3E_2	00	01	11	10
00	0	0	X	1
01	0	X	X	X
11	X	X	X	X
10	1	X	X	X

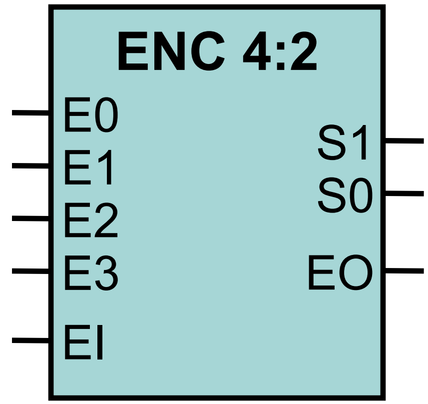
$$S_0 = E_1 + E_3$$

Priority Encoders

- Characteristics
  - Prioritize multiple input activation
  - Priority:
    - To the most significant bit (MSB): priority is given to the MSB
      E1 and E5 active, result is 5
    - To the least significant bit (LSB): priority is given to the LSB
      E1 and E5 active, result is 1

Example:
4:2 MSB priority encoder

- M:N ⇒ 'M' inputs, 'N' outputs
- EO: "Enable Output"
- EI ó E: "Enable Input" o "Enable"
  - Enables the circuit:
    - '0' (disabled): outputs are tied to '0'
    - '1' (enabled): normal operation
  - EI and OE are used together to chain encoders.



EI	E_3	E_2	E_1	E_0	S_1	S_0	EO
0	X	X	X	X	0	0	0
1	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	X	0	1	0
1	0	1	X	X	1	0	0
1	1	X	X	X	1	1	0

Example:
4:2 MSB priority encoder

EI	E_3	E_2	E_1	E_0	S_1	S_0	EO
0	X	X	X	X	0	0	0
1	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	X	0	1	0
1	0	1	X	X	1	0	0
1	1	X	X	X	1	1	0

- Reminder
  - 'X' output ⇒ 'X' in table
  - 'X' input ⇒ several cases

E_3E_2 \ E_1E_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$$S_1 = EI(E_2 + E_3)$$

E_3E_2 \ E_1E_0	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

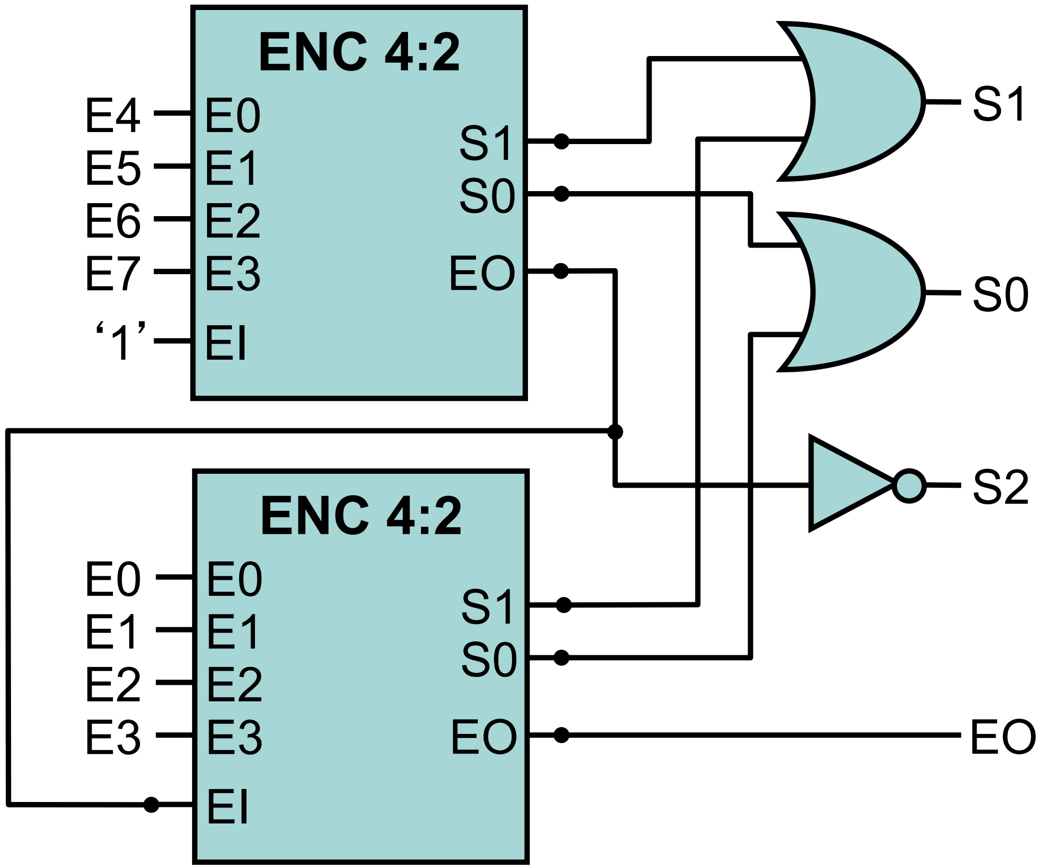
$$S_0 = EI(E_1\overline{E_2} + E_3)$$

E_3E_2 \ E_1E_0	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

$$EO = EI(\overline{E_3}\,\overline{E_2}\,\overline{E_1}\,\overline{E_0})$$

Encoder chains: ENC 8:3 using two ENC 4:2

- EI and EO are chained
- When upper ENC is active (EI= '1') and it has no active inputs, it enables lower ENC (EO= '1').

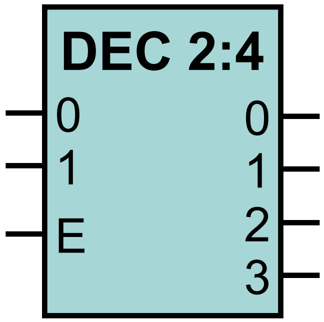


Encoder utility

- Elevator sensors
  - Encode each sensor to the floor number
  - No priority needed. The elevator can activate only one sensor.
- Keypad
  - Encodes the value of the pressed key
  - Priority is required. Several keys may be pressed at the same time.

2. Decoders

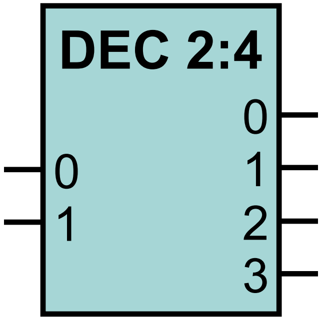
- Definition:
  - Converts an encoded value, activating the output that corresponds to the converted value
  - Function opposite to encoders



E	E_1	E_0	S_3	S_2	S_1	S_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Decoders

- Logic functions:
  - Each output is a minterm

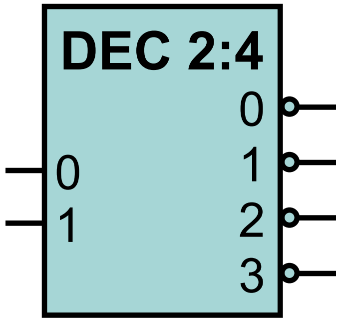


E_1	E_0	S_3	S_2	S_1	S_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$$S_0 = \overline{E_1}\,\overline{E_0}$$
$$S_1 = \overline{E_1} E_0$$
$$S_2 = E_1 \overline{E_0}$$
$$S_3 = E_1 E_0$$

Decoders

- Low-level output decoders:
  - Each output is a maxterm



E_1	E_0	S_3	S_2	S_1	S_0
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

$$S_0 = E_1 + E_0$$

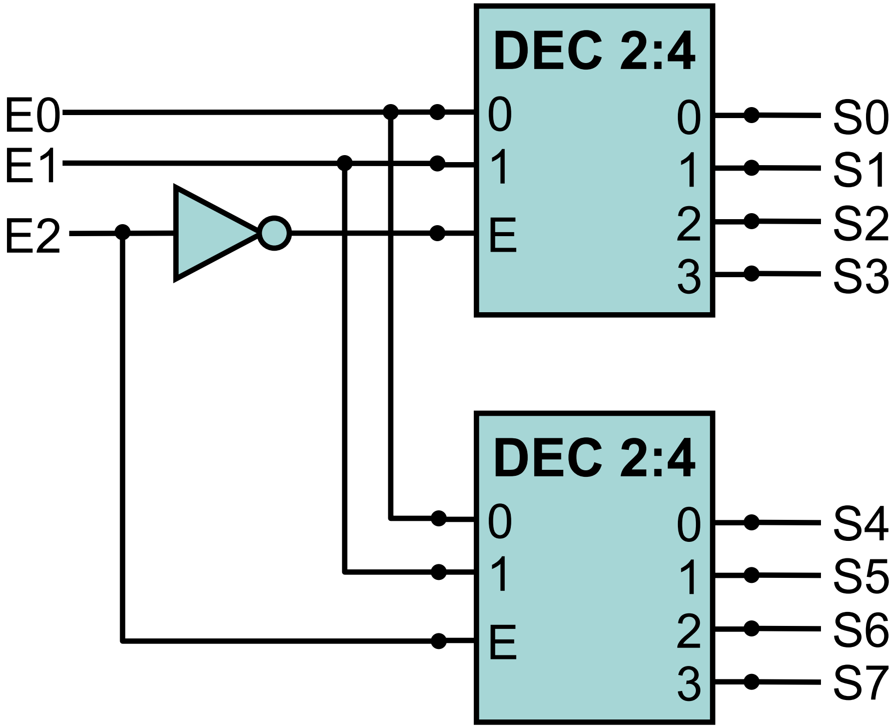
$$S_1 = E_1 + \overline{E_0}$$

$$S_2 = \overline{E_1} + E_0$$

$$S_3 = \overline{E_1} + \overline{E_0}$$

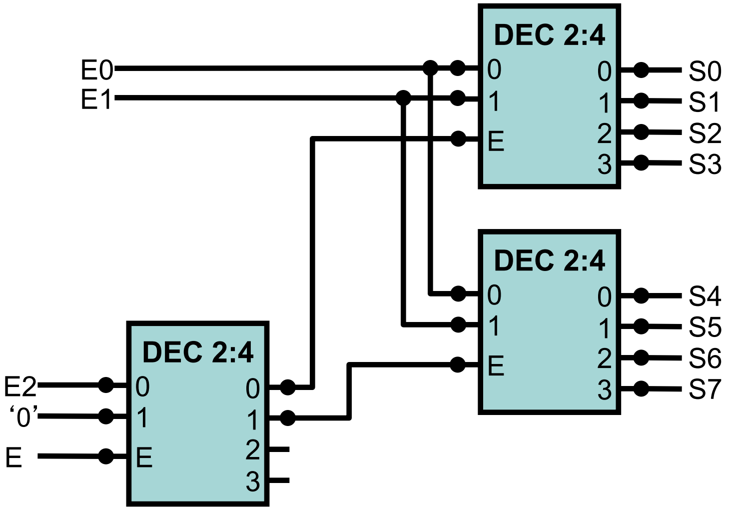
Decoder association

- DEC 3:8 with two DEC 2:4
  - Just one decoder is active, depending on E2 value
  - The inverter behaves as a DEC 1:2
  - Doesn't have an Enable input



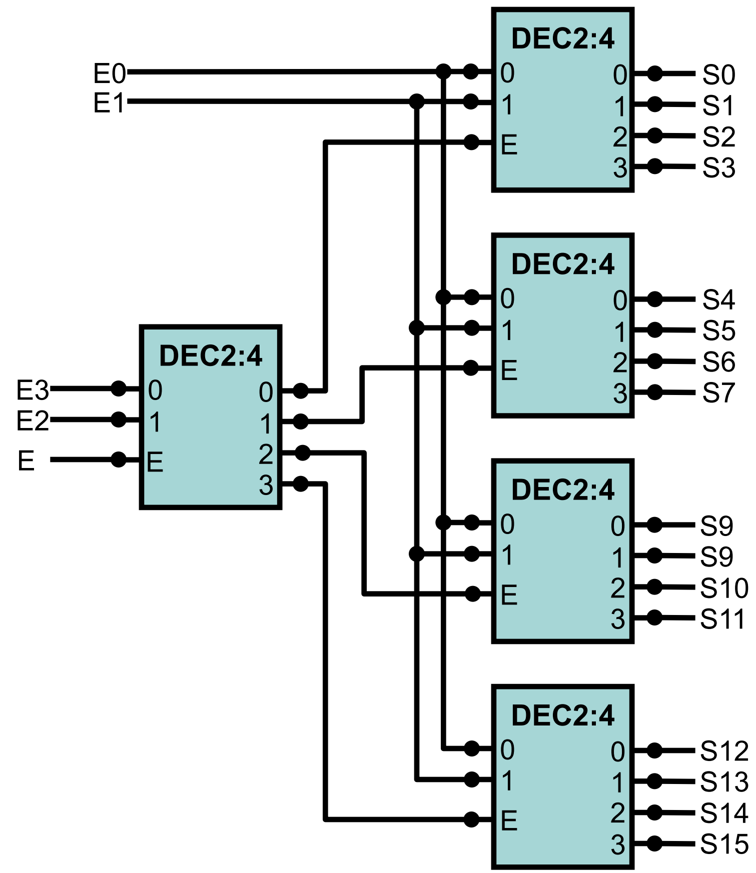
Decoder association

- DEC 4:16 with DEC 2:4
  - Just one decoder (on the right) is active, depending on E2 value
  - Left decoder behaves like a DEC1:2
  - There is a Global Enable. If E= ‘0’ , no decoders are active and all outputs are null



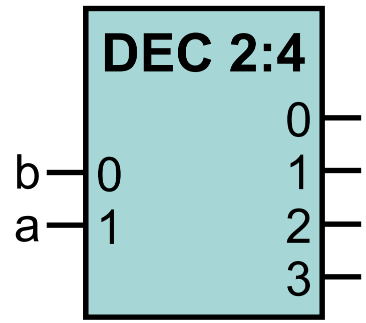
Decoder association

- DEC4:16 with DEC2:4
  - Just one decoder (on the right) is active, depending on E2 and E3 values



Logic function implementation using decoders

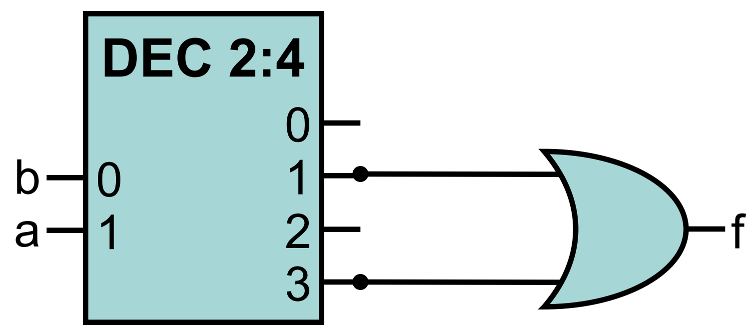
- Logic functions can be implemented using a decoder and an OR gate
- DEC outputs are minterms. Outputs with ‘1’ value in truth table are Ored
- Dual case is built with a low-level output DEC and an AND gate



$$S_0 = \bar{a}\bar{b}$$
$$S_1 = \bar{a}b$$
$$S_2 = a\bar{b}$$
$$S_3 = ab$$

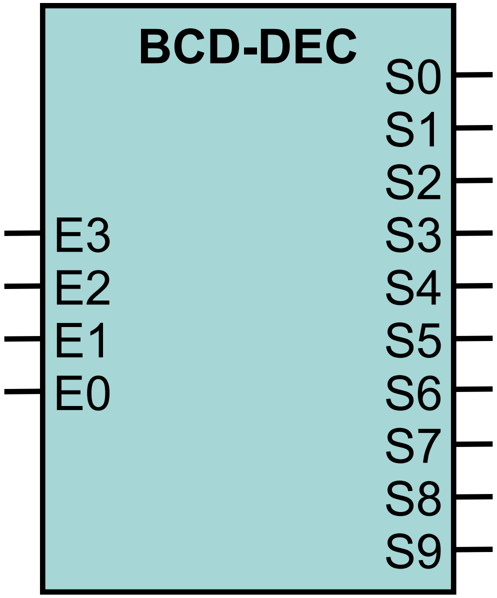
m	a	b	f
0	0	0	0
1	0	1	1
2	1	0	0
3	1	1	1

$$f = \bar{a}b + ab = S_1 + S_3$$



BCD-decimal decoder

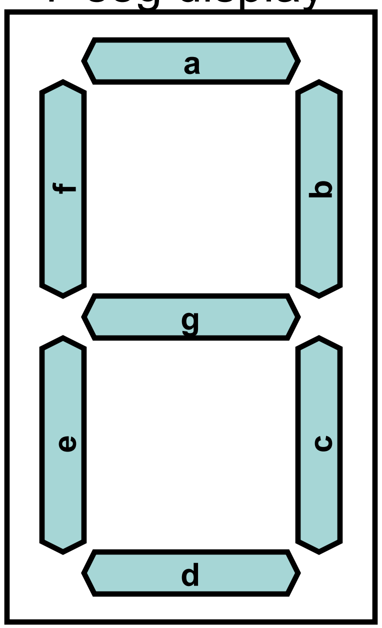
- Decodes a BCD encoded decimal digit. 10 outputs (0-9)
- Behaviour is undefined if input is not decimal



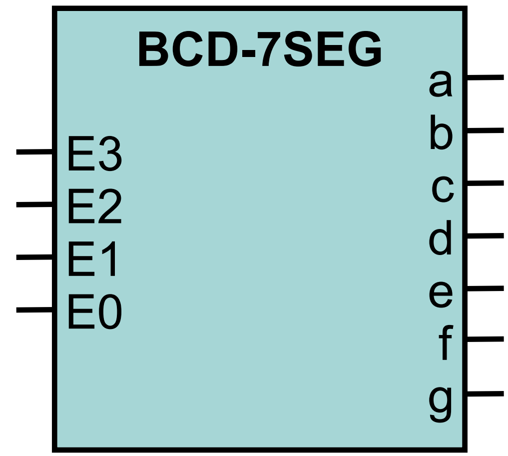
BCD-7segment decoder

- Decodes a BCD encoded decimal digit to the LEDs of a 7-segment display

7-seg display



E_3	E_2	E_1	E_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
Others				X	X	X	X	X	X	X

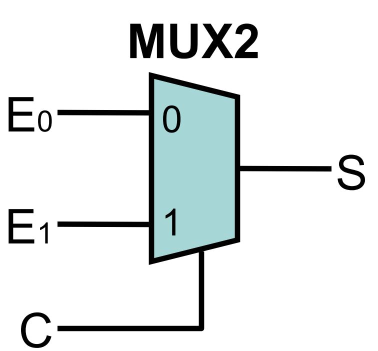


Utility of decoders

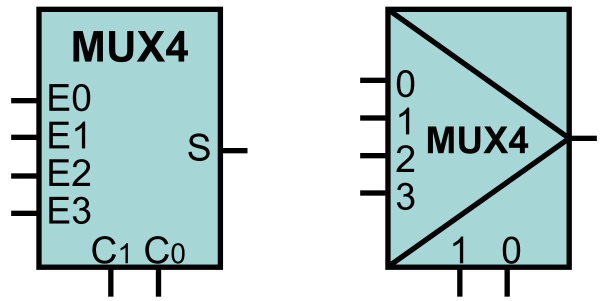
- Microprocessors:
  - Instruction decoder
  - I/O port address, memory address.

3. Multiplexers

- Definition:
  - Combinational circuit that selects one input according to a control signal and copies its value to the output
  - Named by the number of data inputs: MUX2, MUX4, …
  - N=data inputs, n=control inputs ⇒ $2^n = N$



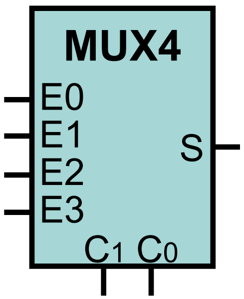
C_0	S
0	E_0
1	E_1



C_1	C_0	S
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

Multiplexers

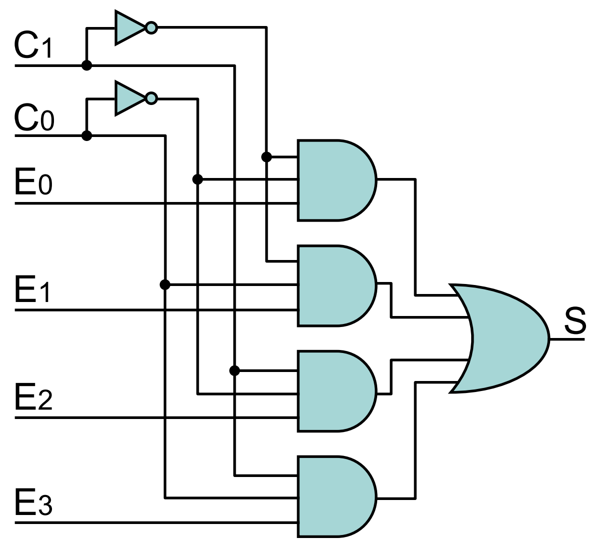
- Logic function



C_1	C_0	S
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

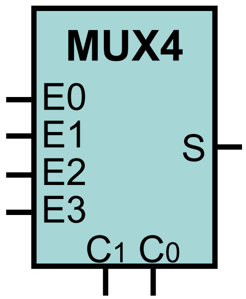
$$S = \overline{C_1}\,\overline{C_0}E_0 + \overline{C_1}C_0E_1 + C_1\overline{C_0}E_2 + C_1C_0E3_0$$

- Implementation with gates



Multiplexers

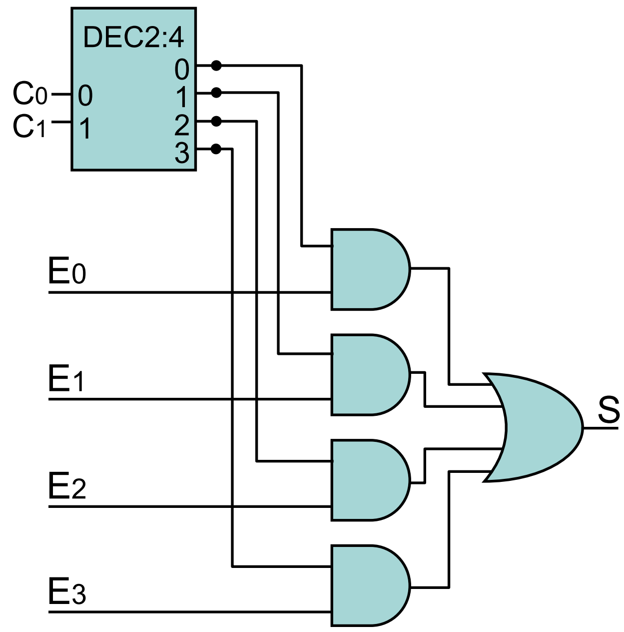
- Logic function



C_1	C_0	S
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

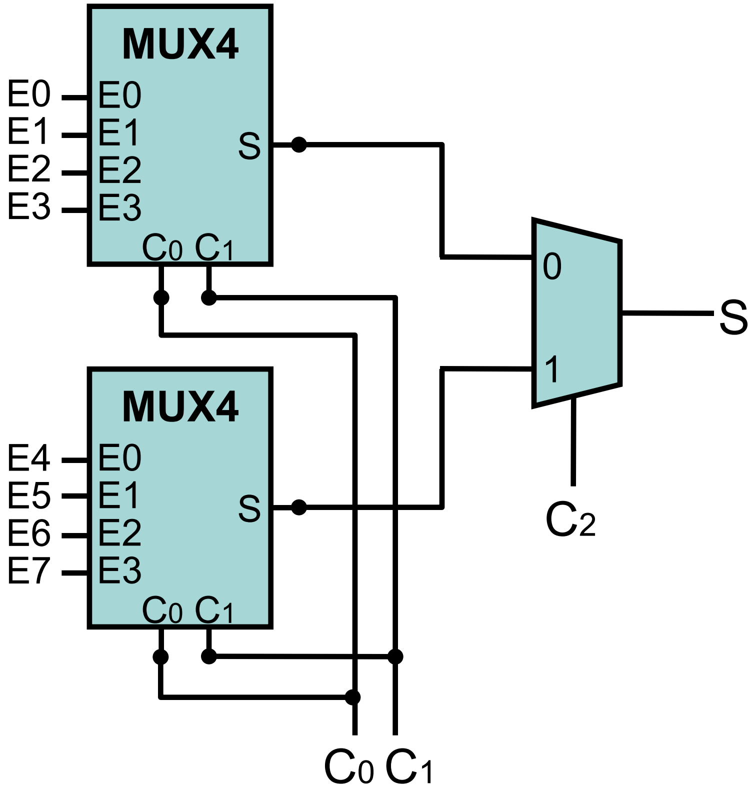
$$S = \overline{C_1}\,\overline{C_0}E_0 + \overline{C_1}C_0E_1 + C_1\overline{C_0}E_2 + C_1C_0E3_0$$

- Implementation with a decoder



Multiplexer association

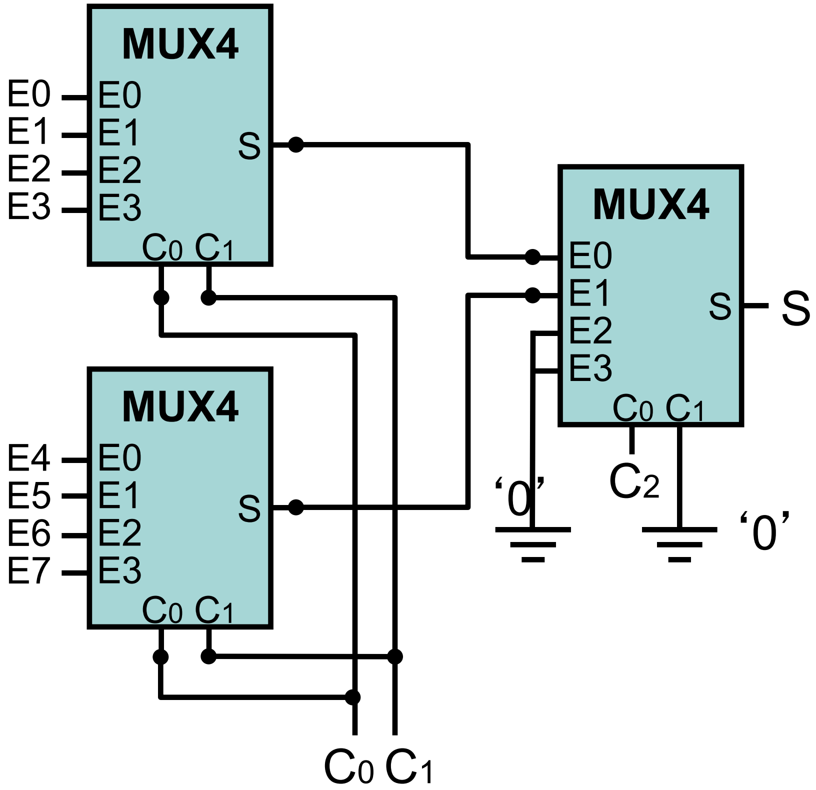
- MUX8 with MUX4 and MUX2



- Right MUX2 selects between upper and lower MUX4 depending on the most significant control bit (C_2)
- C and E bits must be assigned according to their weigh

Multiplexer association

- MUX8 with MUX4

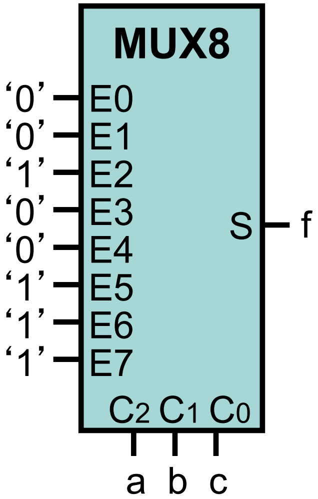


- Right MUX4 behaves as a MUX2
- Remember: circuit inputs MUST have a value, while outputs may be left unconnected

Logic function implementation using multiplexers

- Using a MUX with as many control inputs as function variables
  - Function variables are connected to MUX control inputs, weigh sorted
  - Values in the truth table are assigned to MUX data inputs

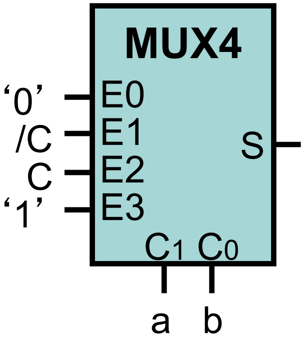
a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Logic function implememtation using multiplexers

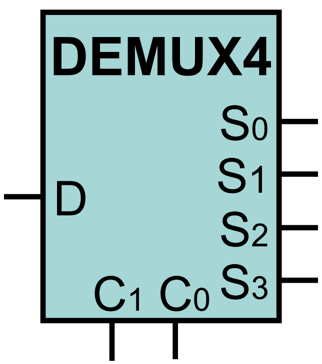
- Using a MUX with less control inputs than function variables
  - Group truth table by least significant variables
  - Most significant variables are connected to MUX control inputs, weight sorted
  - Values in the truth table are assigned to MUX data inputs

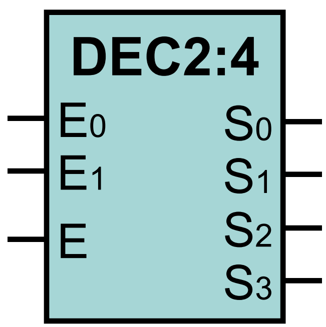
a	b	c	f	f(c)
0	0	0	0	0
0	0	1	0	
0	1	0	1	/C
0	1	1	0	
1	0	0	0	C
1	0	1	1	
1	1	0	1	1
1	1	1	1	



4. Demultiplexers

- Definition: combinational circuit that copies the input to one of the outputs, depending on the values of the control signals
- Opposite multiplexers behaviour
- Equivalent to decoders, if DEMUX control (C_i) are DEC data (E_i) and DEMUX data (D) is DEC enable (E)

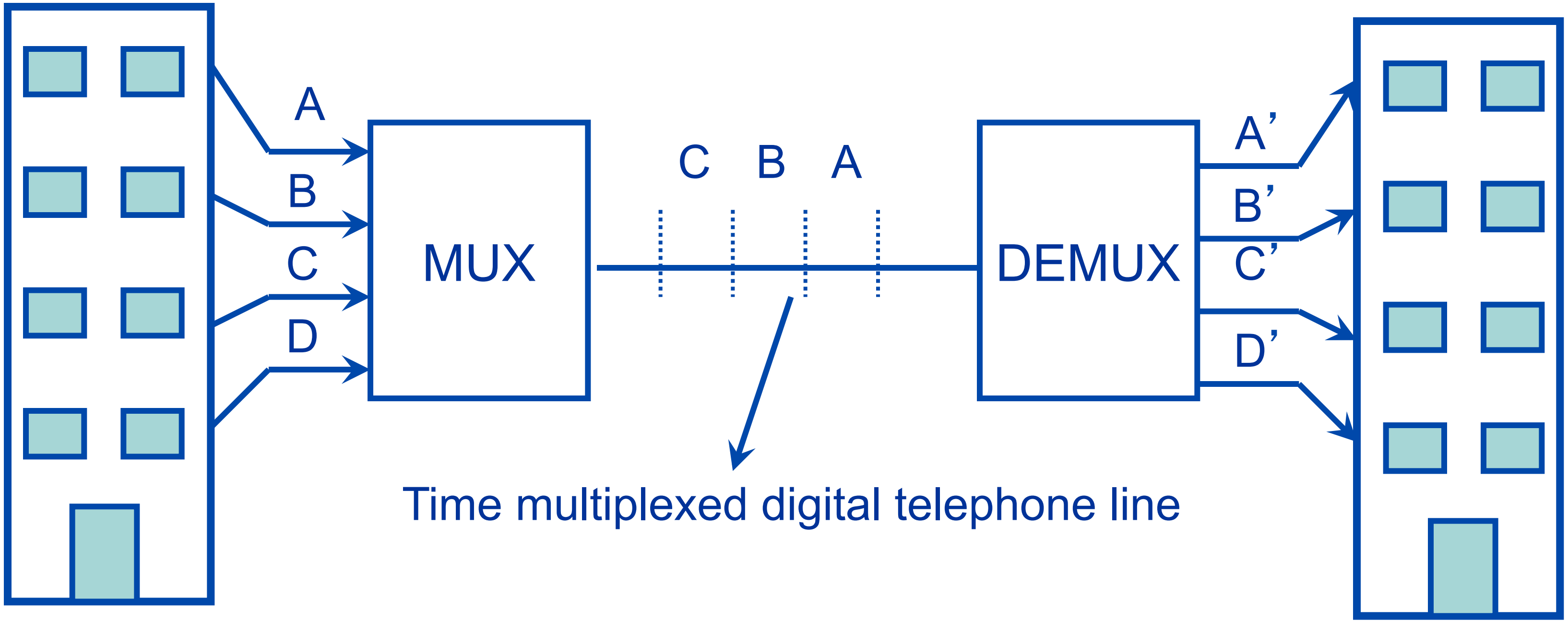




D	C_1	C_0				
E	E_1	E_0	S_3	S_2	S_1	S_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

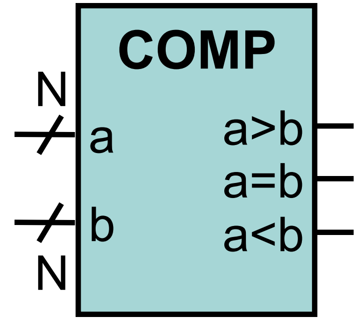
Utility of multiplexers and demultiplexers

- Multiplexed serial transmision



5. Comparators

- Definition: combinational circuit that determines if a number is greater than, equal to or lower than other
- N-bit data



1-bit comparator

a	b	a=b	a>b	a<b
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

$$f_{a=b} = \overline{a \oplus b}$$

$$f_{a>b} = a\overline{b}$$

$$f_{a<b} = \overline{a}b$$

Comparators

- 3-bit comparator

$$f_{a=b} = \overline{(a_2 \oplus b_2)} \cdot \overline{(a_1 \oplus b_1)} \cdot \overline{(a_0 \oplus b_0)}$$

$a_2=b_2$ $a_1=b_1$ $a_0=b_0$

$$f_{a>b} = a_2\overline{b_2} +$$ $a_2>b_2$

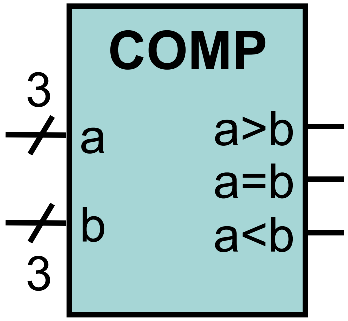
$$+ \overline{(a_2 \oplus b_2)} \cdot a_1\overline{b_1} +$$ $a_2=b_2$ and $a_1>b_1$

$$+ \overline{(a_2 \oplus b_2)} \cdot \overline{(a_1 \oplus b_1)} \cdot a_0\overline{b_0}$$ $a_2=b_2$, $a_1=b_1$ and $a_0>b_0$

$$f_{a>b} = \overline{a_2}b_2 +$$ $a_2<b_2$

$$+ \overline{(a_2 \oplus b_2)} \cdot \overline{a_1}b_1 +$$ $a_2=b_2$ and $a_1<b_1$

$$+ \overline{(a_2 \oplus b_2)} \cdot \overline{(a_1 \oplus b_1)} \cdot \overline{a_0}b_0$$ $a_2=b_2$, $a_1=b_1$ and $a_0<b_0$



- It can be generalized
- High gate use (XOR)

Bibliografía

- “Circuitos y Sistemas Digitales”. J. E. García Sánchez, D. G. Tomás, M. Martínez Iniesta. Ed. Tebar-Flores
- “Electrónica Digital”, L. Cuesta, E. Gil, F. Remiro, McGraw-Hill
- “Electric circuits fundamentals”, T.L Floyd, Prentice-Hall