عال ٢٢٠ جزئية الميد

Unit 1 Numbers Systems

System	decimal	Binary	Octal	Hexa
Base	10	2	8	16
Symbols	0 1 2 3 4 5 6 7 8 9	0 1	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7 8 9 A B C D E F

Decimal	Binary	Octal	Hexa-decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17

للتحويل من أي نظام إلى النظام العشري :. أستخدم قاعدة "Base & Weight"

- فكرة الـ Weight عبارة عن أس يعبر عن الخانة الموجود فيها الرقم.
- خطوات التحويل :.
  - أحدد الـ Base.
  - أضرب الـ Base أس الـ Weight في الخانة المناسبة.
  - مجموع النواتج هي قيمتها بـ decimal.
- مثال :.

من Hex الى dec

ABC_{16}

$C \times 16^0 = 12 \times 16^0 = 12$
$B \times 16^1 = 11 \times 16^1 = 176$
$A \times 16^2 = 10 \times 16^2 = 2560$
2748_{10} الناتج dec بـ

من Oct الى dec

724_8

$4 \times 8^0 = 4$
$2 \times 8^1 = 16$
$7 \times 8^2 = 448$
468_{10} الناتج dec بـ

من Bi الى dec

101011_2

$1 \times 2^0 = 1$
$1 \times 2^1 = 2$
$0 \times 2^2 = 0$
$1 \times 2^3 = 8$
$0 \times 2^4 = 0$
$1 \times 2^5 = 32$
43_{10} الناتج dec بـ

للتحويل من النظام العشري إلى أي نظام :. أقسم على الـ Base المراد التحويل له

- فكرة التحويل بعد القسمة أوجد باقي القسمة لانه سيصبح الناتج .
- أول باقي قسمة سيصبح أول خانة على اليمين أو مايسمى بـ LSB - Least Significant bit .
- لو كانت ارقام عشرية يصبح ضرب .
- نتوقف عن القسمة عند الوصول لناتج 0.
- مثال :.

من dec الى Bi

125_{10}

	الباقي	
125/2 = 62	1	LSB
62/2 = 31	0	
31/2 = 15	1	
15/2 = 7	1	
7/2 = 3	1	
3/2 = 1	1	
1/2 = 0	1	

1111101_2

للتحويل من نظام إلى نظام آخر غير النظام العشري :.

Oct ⟶ Bi

اخذ كل خانة من Oct واشوف أيش يكافئها بالـ Bi
* لازم تكون ٣ وحدات الـ Bi

مثال :. 705_8

7 0 5
111 000 101

111000101_2

Hex ⟶ Bi

اخذ كل خانة من Hex واشوف أيش يكافئها بالـ Bi
* لازم تكون ٤ وحدات الـ Bi

$10AF_{16}$

1 0 A F
0001 0000 1010 1111

0001000010101111_2

Bi ⟶ Oct

من اليمين ناخذ كل ٣ وحدات من Bi وندشوف المكافئ لها في Oct

1011010111_2

001 011 010 111
1 3 2 7

1327_8

Bi ⟶ Hex

من اليمين ناخذ كل ٤ وحدات من Bi وندشوف المكافئ لها في Hex

10101011_2

0010 1011 1011
2 B B

$2BB_{16}$

عند التحويل بين Hex و Oct يجب أن نحول الـ Bi أولاً

نظام Binary coded Decimal أو BCD :

the 4-bit BCD is usually employed by the computer to represent and process numircal data.

Example : 5327_{10} in BCD

4-bits of 5 ⟶ 0101
4-bits of 3 ⟶ 0011
4-bits of 2 ⟶ 0010
4-bits of 7 ⟶ 0111

Therfore the BCD of 5327 is

0011001000111

Unit 2 Digital circuit Design

Digital curcit

(hardware) manipulate binary information.

↓ Consists of few simple building blocks called logic gates.

→ logic gates

an electronic device that operates on 1 or more input signals and produce an output.

- Basic gates: (AND, OR, NOT) ← الأولوية في الحالات
  1- NOT
  2- AND
  3- OR
- Additional gates: (NAND, NOR, XOR, XNOR)
  - NAND = NOT + AND
  - NOR = NOT + OR
  - XOR = exclusive OR :: لازم تكون المدخلات مختلفة يعني ١و٠ لو نفس بعض بيصير المخرج 0 — ⊕ → $X = A \oplus B$, $X = \bar{A}B + A\bar{B}$
  - XNOR = complement of XOR — لازم تكون المدخلات متشابهه يعني 0و0 أو 1و1 مختلف = 0 → $X = AB + \bar{A}\bar{B}$

• logic gates are built using transistors.
- NOT : implemnted by 1 transistor
- AND-OR : requires 3 transistor.

⟹ Trasistors are Fundamental
- Pentium cosists of 3 million transistors
- Compaq Alpha cosists of 9 million transistors
- Now we build chips with more than 100M

→ logic function

الدالة المنطقية أقدر أعبر عنها بـ٣ أشكال
1- truth table
2- logical expression
3- logic Digram

o A Boolean Function consist of
1- Binary variables
2- constants 0,1
3- logic oprators (•)(+)(-)

a Function with n inputs will have 2^n combination inputs

a single-output function relates the output (0/1) to inputs.

Multiple-output boolean function has more than 1 output and each output (0/1) related to same input.

→ Standard forms

SOP : sum of products

- نأخذ من الجدول القيم اللي تساوي 1
- $B'C + ABC$ ← مضروبات مجموعه، زي ما أحب اسميها :|
- نستخدم في الدائرة And-Or
- نستخدم two-level circut — 2 و 1 و 0 هذا level خاص بالنفي
- الصفوف فيه تسمى minterms واللي هي عبارة عن مضروب المدخلات مع نفي *
- عدد minterms يساوي 2^n حيث ان n تعبر عن عدد المتغيرات
- Sum of minterms: is a special kind of sum of Products — اقدر اسويه في أي داله
- Canonical SOP : يحدث عندما تكون كل النواتج = 1

إثراء رياضي:
ليش ناخذ اللي يساوي 1 ؟
لان دالة SOP عبارة عن
$$F = \sum (m_i \cdot f_i)$$
m_i: رقم minterm — f_i: ناتج الصف
علشان كذا لما يكون المخرج 0 ما يطلع رقم minterm

راجعوا الجدول أول صف كل التغيرات منفية

POS : Product of sums

- نأخذ من الجدول القيم اللي تساوي 0
- $(B'+C)(B+A+C)$ ← مجموعات مضروبه :|
- نستخدم في الدائرة Or - And
- نستخدم two-level circut — 2 و 1 و 0 هذا level خاص بالنفي
- الصفوف فيه تسمى maxterms واللي هي عبارة عن مضروب المدخلات مع نفي *
- عدد maxnterms يساوي 2^n حيث ان n تعبر عن عدد المتغيرات
- Product of maxterms: is a special kind of Product of sums — اقدر اطبقه باي داله
- Canonical SOP : يحدث عندما تكون كل النواتج = 0

إثراء رياضي:
ليش ناخذ اللي يساوي 0 ؟
لان دالة POS عبارة عن
$$F = \prod (m_i \cdot f_i')$$
m_i: رقم maxterm — f_i': نفي ناتج الصف
علشان كذا لما يكون المخرج 1 ما يطلع رقم maxterm

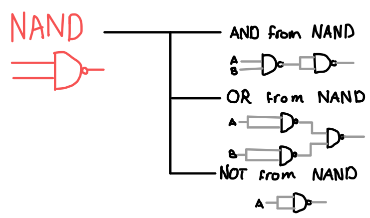
راجعوا الجدول أخر صف كل التغيرات منفية

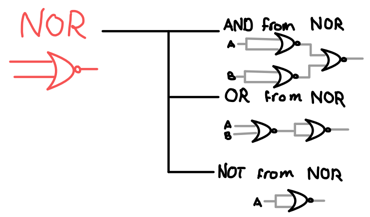
Designing a logic circuit

- truth table is used to represent a logic function
- logic expressions can be obtained from truth table
- logic expressions can be transfer to logic Digram

Universal gates

سميت universal لأني أقدر أكون منها كل البوابات الرئيسية (NOT-AND-OR)





Logical Equivalence

إذا كان فيه دائرتين مختلفة يطبقون نفس الدالة
نقدر نطلع التكافؤ عن طريق
- trace circut path
- truth table

Logic chips Integration level

- **SSI** — small scale Integration — in late 1960 — 1-10 gates
- **MSI** — medium scale Integration — in late 1960 — 10-100 gates
- **LSI** — large scale Integration — in early 1970 — 100-10,000 gates
- **VLSI** — verey large scale Integration — in late 1970 — more than 10,000 gates

Unit 3 Simplification

Karnaugh map can reduce expressions to minimal sum of product or MSP

1. minimal number of product terms
2. minimal number of literals

∘ minimal SOP leads to tow level circut.

- K-map is a matrix consisting of the output of the minterms of a boolean function.

K-map of 3 inputs (y over the m_1, m_3 columns; x on the lower row; z over the m_5, m_7 columns)

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

K-map of 3 inputs (y over the m_1, m_3 columns; x on the lower row; z over the m_5, m_7 columns)

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

K-map of 4 inputs (y over the m_1, m_3 columns; x on the second and third rows; w on the lower two rows; z over the m_5, m_7 columns)

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

خطوات التبسـيط بإستخدام K-map

١- ابدئي بتكوين المعادلة من الجدول

٢- حددي minterms في K-map وعبيها

٣- حددي المجموعات : المجموعات يجب ان تكون من القوى للاثنين وتقدرين تطلعين من طرف المربع للجهه الاخرى

٤- اكتبي المعادلة بعد التبسيط * المجموعه يؤخذ منها المتغيرات المشتركه

٥- حاولي تكون المجموعات اكبر في الحجم واقل في العدد

Don't care

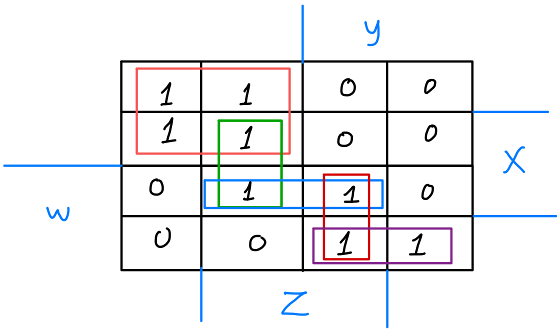
when we don't care what a function output denoted by X in truth table

f(x,w,z) → 1 اللي فيها miterms

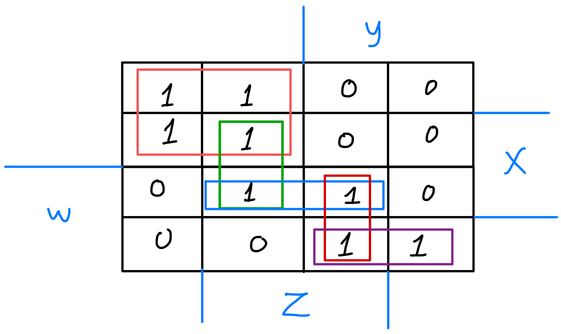
d(x,wz) → X اللي فيها miterms

don't care استخدمها لمصلحتي في K-map لما احتاج اخليها 1 ولما ما احتاج أو راح تزيدني مجوعه راح اخليها 0

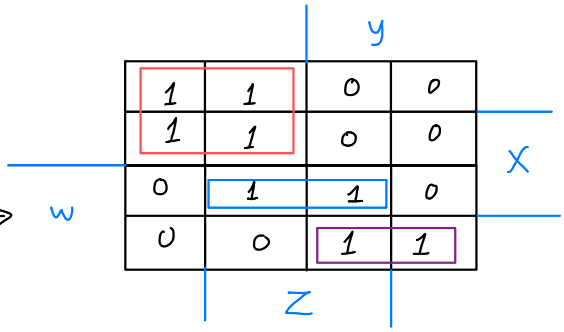
Prime implicant الجموعات الاساسية



Essential Prime implicant الجموعات الضرورية



بالتالي هذه تعتمد لان : غير ضرورية



- ضرورية لان m_0, m_1, m_4 بس موجودين فيها
- ضرورية لان m_{10} بس موجوده فيها

Unit 4 Signed number representation

Unsigned number representation

Tow 1-bit addition

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	10 → two

Tow n-bit addition

$$\begin{array}{r} {}^{1}0{}^{0}0{}^{1}0101 \\ 00011001 \\ \hline 01001101 \end{array} +$$

advantages:

One representation of zero
Simple addition

disadvantages:

Negative numbers can't be represented
The need of diffrent notation to represent negative numbers
you can't just stick a negative sign in front of binary number (it does not work like that 😑)
to represent a negative number

Signed number representation

Signed magnitude: human use that. We add + or - to the front of a number.

we can do this too with binary by adding:
- 0 in front of positive number
- 1 in front of negative number

disadvantage:
- addition and substraction are difficult
- there are two representation of zero 00000, 10000 the CPU will be confused.

One's Complemnt Complementing a bit is equivalent to subtracting it from 1

X	X'	1-X
0	1	1
1	0	0

when complementing n-bit nubmer is equivalent to subtracting that number from 2^n-1

Example: negate 5-bit number 01101

n=5 $2^5-1=11111_2$

$$\begin{array}{r} 11111 \\ -\ 01101 \\ \hline 10010 \end{array}$$

or just negate each bit 😐

There are 2 steps in adding 1's complement
- Do unsigned addition
- include the sign bits

→ Example $\begin{array}{r} 0011 \\ +\ 0010 \\ \hline 0101 \end{array}$ step1 $\quad \begin{array}{r} +0101 \\ 0 \\ \hline 0101 \end{array}$ step2

it's simpiler than Signed addition but still bit tricky

Two's Complemnt the most used representation for integers All +ve number begin with 0/All -ve number begin with 1

to negate a number we complement each bit and then add 1

حركة ثانية : خلي أول رقم عاليمين واعكس الباقي

addition is much easier in 2's complement : add numbers including the sign bit, and ignore carry out.

Example

$$\begin{array}{r} 0111 \\ 1100 \\ \hline \not{1}0011 \end{array}$$

carry out ←

advantage:
One representation of zero
Simple addition
Widely used in ALU

n-bit Representation

We use the $(n-1)^{th}$ bit for sign and remaining bits for magnitude.

Example:

suppose 10011101 is signed magnitude of a 8 bit number.

the sign bit is 1, the magnitude 0011101 with value=29 so -29 in decimal.

More Info:

- positive number are the same in all 3 representation (SM, 1C, 2C)
- in 2C we can represent -8 but not +8.
- 2C more prefered because it has only one 0. and it's addition is the simplest

overflow

The ranges for general n-bit numbers (including the sign bit):

	Unsigned	SM	1C	2C
smallest	0	$-(2^{n-1}-1)$	$-(2^{n-1}-1)$	-2^{n-1}
largest	2^n-1	$+(2^{n-1}-1)$	$+(2^{n-1}-1)$	$+(2^{n-1}-1)$

with 4-bit 2C the largest decimal value is +7 and the smallest is -8

if we try to compute 4+5 or -4 + -5 in 4-bit number it will be overflow

Signed overflow is very diffrent from unsignd overflow the carry out wont be enough to detect overflow

to detect signd overflow look at the sign bits
- if you add 2 +ve numbers and get a -ve result that's overflow
- if you add 2 -ve numbers and get a +ve result that's overflow
- overflow will neve happen when you sum -ve with +ve.

Sign extension

اذا كان الرقم سالب راح نضيف واحدات باليسار على حسب المطلوب
اذا كان الرقم موجب راح نضيف اصفار باليسار على حسب المطلوب

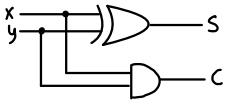
Unit 5 Combinational Circuts-1

Combinational circuts: applying the same input always produce the same outputs.

In programming, combinational circut are similar to "functional programs" that doesn't contain variables and assignments.

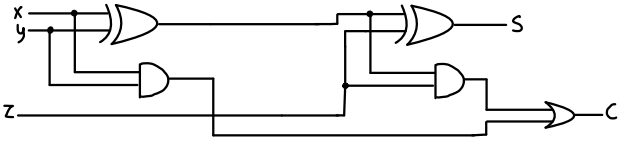
Designing an Adder for 1-bit numbers : 2 inputs (x,y)
2 outputs (C, S) → sum
↓ carry in

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



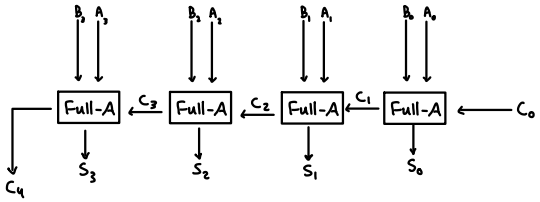
This called half Adder

Full Adder = 2 half Adders

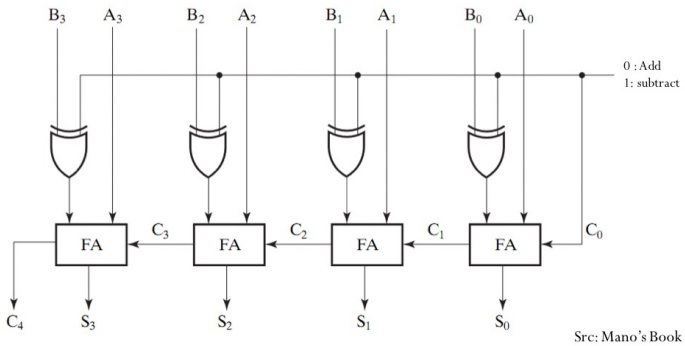


How to build an adder of n-bit numbers?

Example 4-bit Adder
inputs : 9 4-A 4-B 1-carry in
outputs : 5
How many function:5
we use Parallel Full-Adders



This Adder called ripple carry adder



Src: Mano's Book

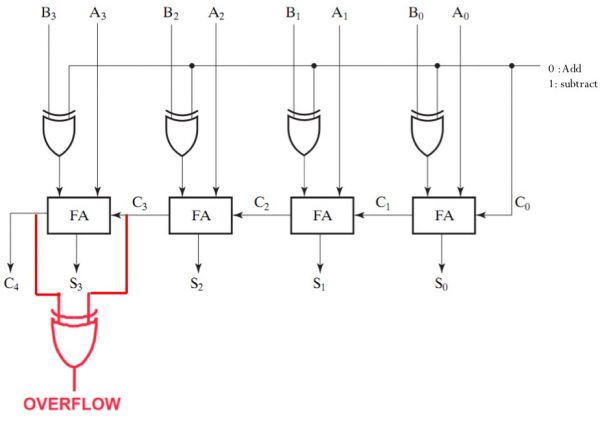
Using full adders and XOR we can build an Adder/Subtractor!

← If you want an adder and a subtractor togther

If you want to detect an overflow →

- Overflow occurs in only 2 situations
  1- if you add 2 +ve numbers and get -ve result.
  2- if you add 2 -ve numbers and get +ve result.

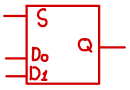
Example : 2 +ve signed numbers



Unit 6 Combinational Circuts-2

Multiplexer:

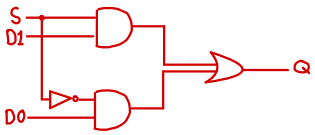
A 2-to-1 multiplexer or mux:



2 data input Do, D1 and a select input called S
1 input named Q
- if S=0, the output will be Do
- if S=1, the output will be D1

S	D1	Do	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$Q = S'D0 + SD1$



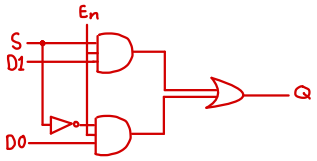
Enable inputs:

many devices have enable input which «activates» or «deactivates» the device.

- EN=0 disable the multiplexer which forces the output to be 0 - but doe'nt trun off the multiplexer
- EN=1 enable the multiplexer

it's usefull with combining small muxes to make large ones.

$Q = En\ S'D0 + En\ SD1$



EN	S	D1	D0	Q
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

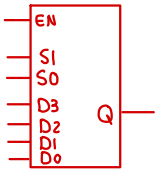
اختصار →

EN	S	D1	D0	Q
0	x	x	x	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

اختصار أكثر →

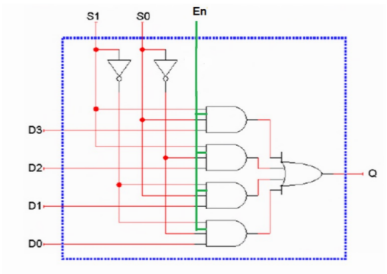
EN	S	Q
0	x	0
1	0	D0
1	1	D1

A 4-to-1 multiplexer or mux:



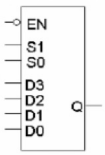
EN	S1	S0	Q
1	0	0	D0
1	0	1	D1
1	1	0	D2
1	1	1	D3
0	x	x	0

$Q = En\ S1'S0'D0 + En\ S1'S0\ D1 + En\ S1S0'D2 + En\ S1S0\ D3$



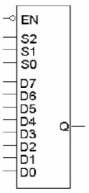
Active-low

LogicWorks multiplexer have active-low enable inputs so the mux always outputs 1 when EN'=1

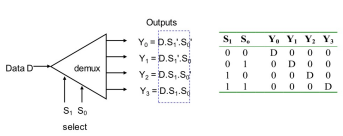


EN'	S1	S0	Q
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	x	x	1

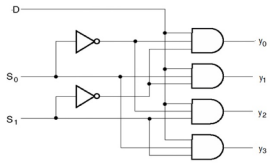
8-1 multiplexer:



Demultiplexer: عكس الـ MUX



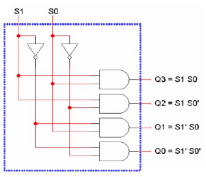
S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D



Decoder: A n-to-2^n decoder uses n-bit to determine which of 2^n outputs will be uniquely active

S1	S0	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Q0 = S1'S0'
Q1 = S1'S0
Q2 = S1 S0'
Q3 = S1 S0



→ next Page for more ...

Enable in decoder:

just as with mux, decoders can include enable inputs

-EN=0 disable the decoder - all outputs =0-

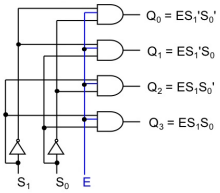
-EN=1 enable the decoder

EN	S1	S0	Q0	Q1	Q2	Q3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

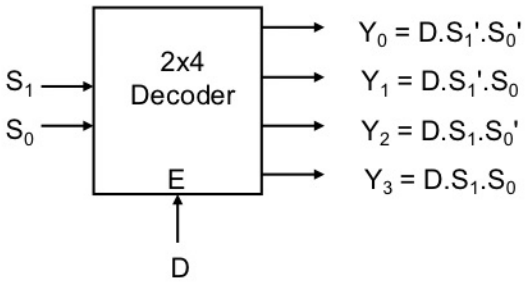
Truth table:

E	S_1	S_0	Q_0	Q_1	Q_2	Q_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	X	X	0	0	0	0

- Circuit:

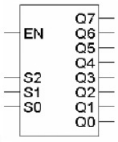


Decoder Vs demux: The demultiplexer is actully identical to a decoder with enable



A 3-to-8 decoder

↓ S0 S1 S2 ↓ Q0-Q7



S2	S1	S0	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Q0 = S2'S1'S0'
Q1 = S2'S1'S0
Q2 = S2'S1 S0'
Q3 = S2'S1 S0
Q4 = S2 S1'S0'
Q5 = S2 S1'S0
Q6 = S2 S1 S0'
Q7 = S2 S1 S0

What good is a decoder ?

decoders sometimes called minterms generators

we can use decoder implement sum of minterms expression

when S2=0 , Q0,Q3 are 2-to-4 decoder

when S2=1 , Q4,Q7 are 2-to-4 decoder

Encoder:

عكس الـ Dec

2^n inputs or less an n outputs.

so inputs are minterms

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

المراجعة :

- الـ Mux يختار من بطلع من المدخلات بناءً على قيمة الـ S .
- الـ Mux دائماً بطلع مخرج واحد وعدد المدخلات والـ S كالتالي لو كان 8X1 ← 2^3 X1 (عدد المدخلات ← عدد S).
- الـ demux عكس الـ Mux فبالتالي عندي فقط مدخل واحد وعدد الـ S بناءً على المخرجات 1X8 ← 1X2^3 (عدد S).
- الـ Dec يفعل أحد المخرجات بناءً على المدخلات ، يفعل يعني يخلي أحد المخرجات 1*.
- الـ Dec متاح من $n-2^n$ والـ Enc عكسه 2^n-n .

- دعواتكم .