14.10.20 12:05

CSCI 150
Introduction to Digital and Computer System Design
Lecture 3: Combinational Logic Design III

Jetic Gū

2020 Fall Semester (S3)

Overview

- Focus: Logic Functions
- Architecture: Combinatory Logical Circuits
- Textbook v4: Ch3 3.6; v5: Ch3 3.1, 3.4
- Core Ideas:
    1. Terminologies: Value-Fixing, Transferring, Inverting, Enabler
    2. Decoder

P0
Review

Systematic Design Procedures

1. **Specification**: Write a specification for the circuit

2. **Formulation**: Derive relationship between inputs and outputs of the system e.g. using truth table or Boolean expressions

3. **Optimisation**: Apply optimisation, minimise the number of logic gates and literals required

4. **Technology Mapping**: Transform design to new diagram using available implementation technology

5. **Verification**: Verify the correctness of the final design in meeting the specifications

Review

P0
Review

Systematic Design Procedures

- Hierarchical Design
  - Divide complex designs into smaller functional blocks, then apply the same 5-step design procedures for each block
  - Reusable, easier and more efficient Implementation

Review

Value-Fixing, Transferring, Inverting, Enabler

Elementary Combinational Logic Functions

Summary

P1
Elementary Func.

Value-Fixing, Transferring, and Inverting

① **Value-Fixing**: giving a constant value to a wire

- $F = 0; F = 1;$

② **Transferring**: giving a variable (wire) value from another variable (wire)

- $F = X;$

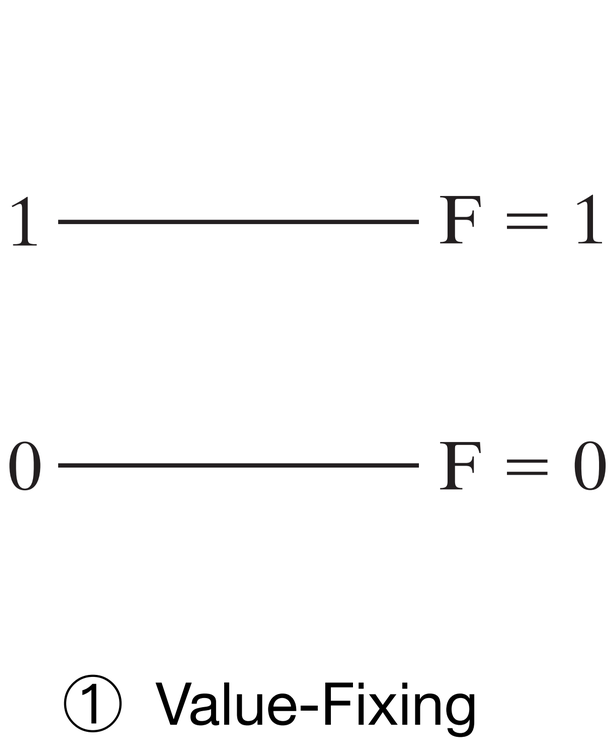
③ **Inverting**: inverting the value of a variable

- $F = \overline{X}$

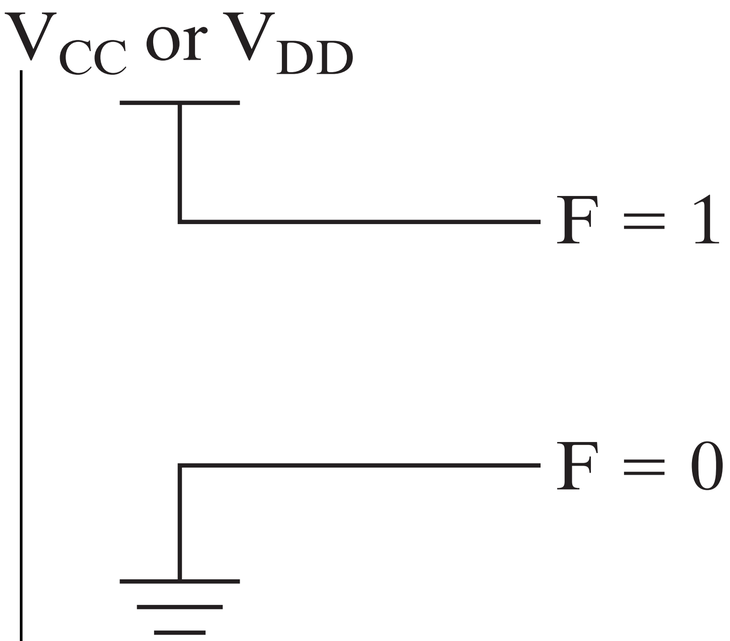
Concept

P1
Elementary Func.

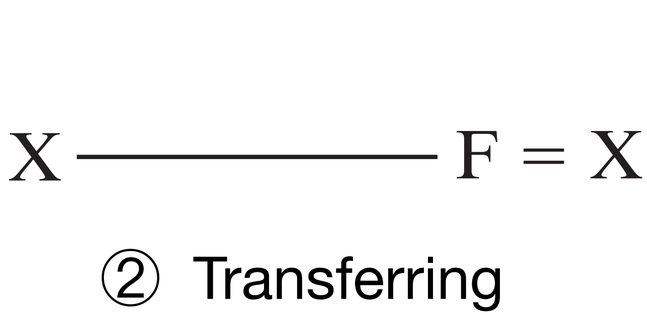
Value-Fixing, Transferring, and Inverting



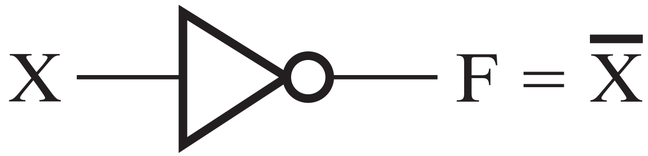
① Value-Fixing



① Value-Fixing



② Transferring



③ Inverting

Concept

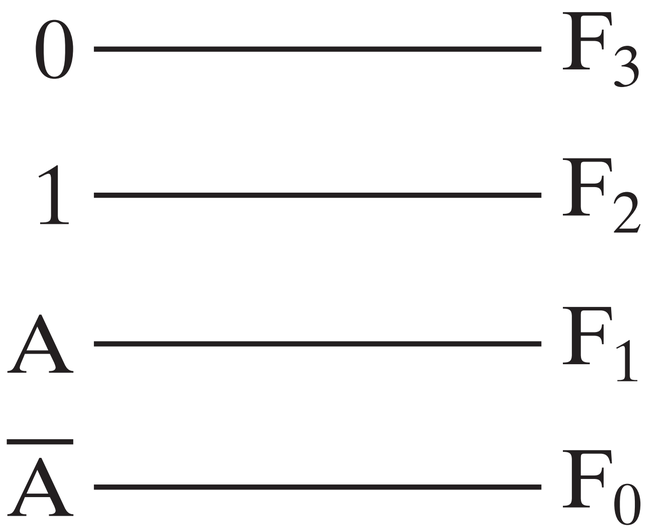
Vector Denotation

④ Multiple-bit Function

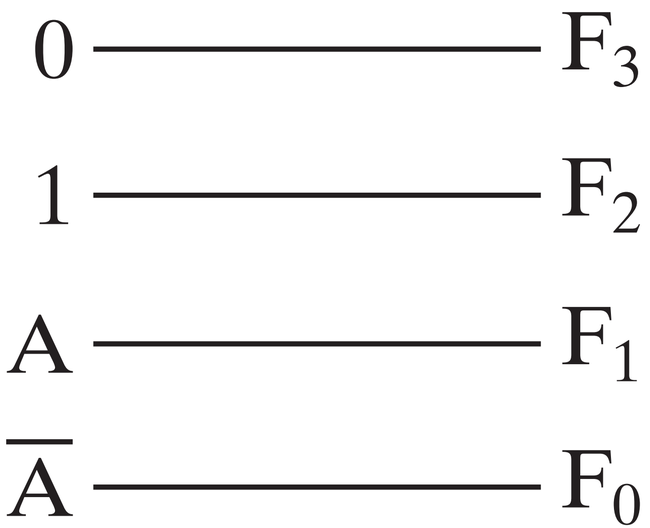
- Functions we've seen so far has only one-bit output: 0/1
- Certain functions may have n-bit output
  - $F(n-1:0) = (F_{n-1}, F_{n-2}, \ldots, F_0)$, each F_i is a one-bit function
  - Curtain Motor Control Circuit: $F = (F_{\text{Motor}_1}, F_{\text{Motor}_2}, F_{\text{Light}})$

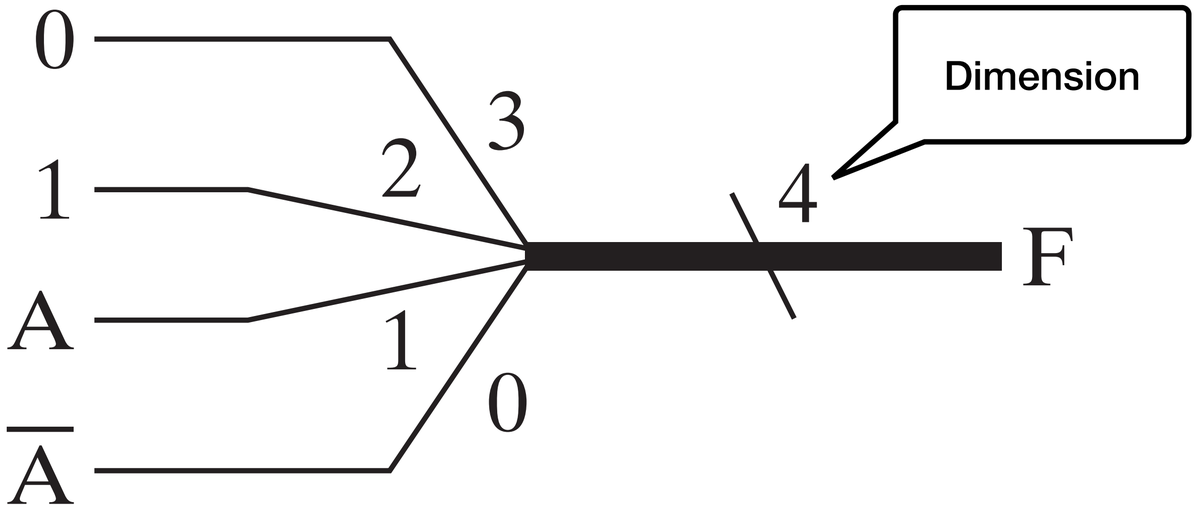
Concept

Vector Denotation



Vector Denotation





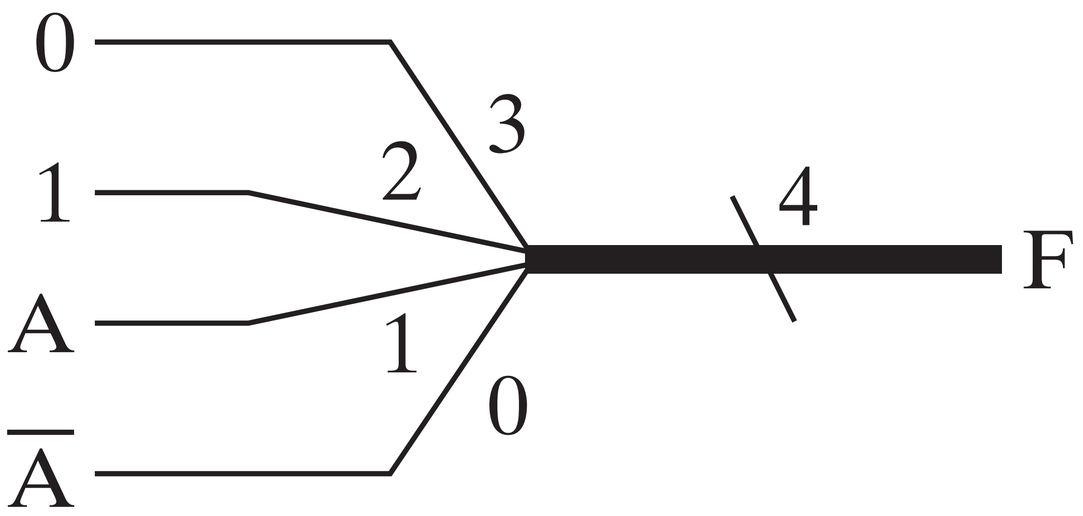
④ Multiple-bit Function

1. These two are equivalent

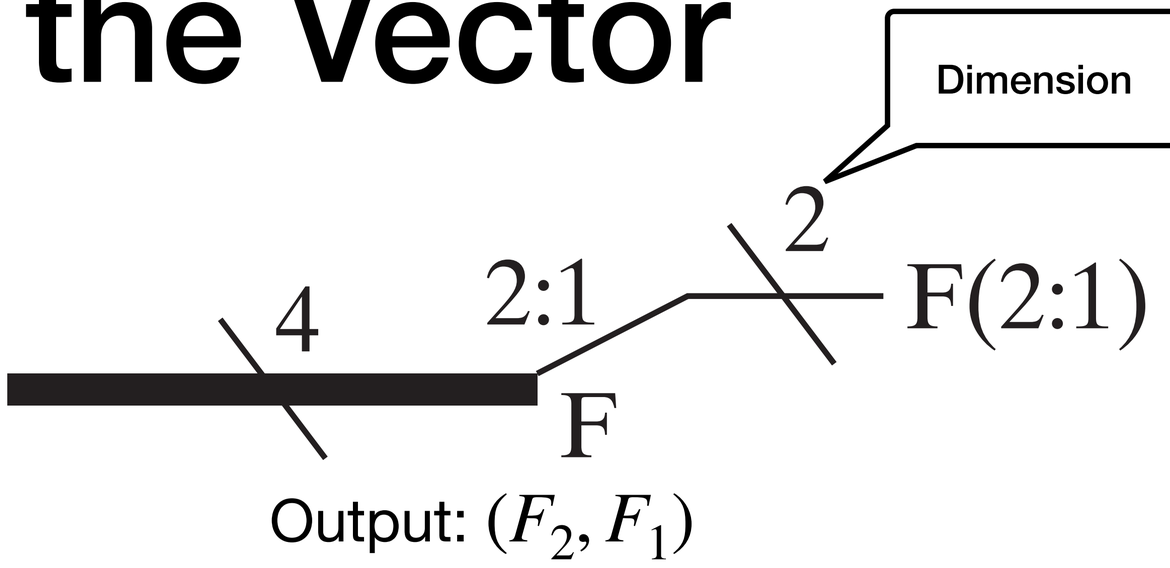
Concept

P1
Elementary Func.

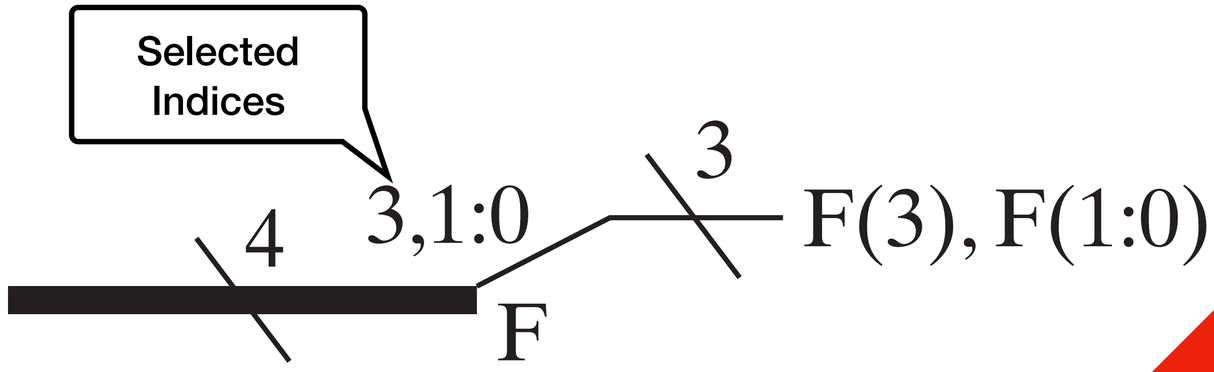
Taking part of the Vector



④ Multiple-bit Function



Output: (F_2, F_1)



Output: (F_3, F_1, F_0)

Concept

Enabler

⑤ Enabler

- Transferring function, but with an additional EN signal acting as switch

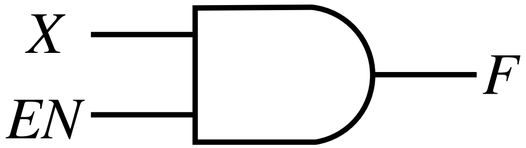
EN	X	F
0	X	0
1	0	0
1	1	1

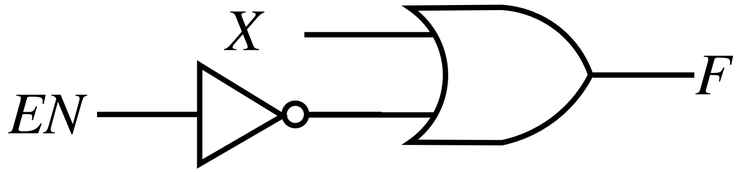
Concept

Enabler

⑤ Enabler

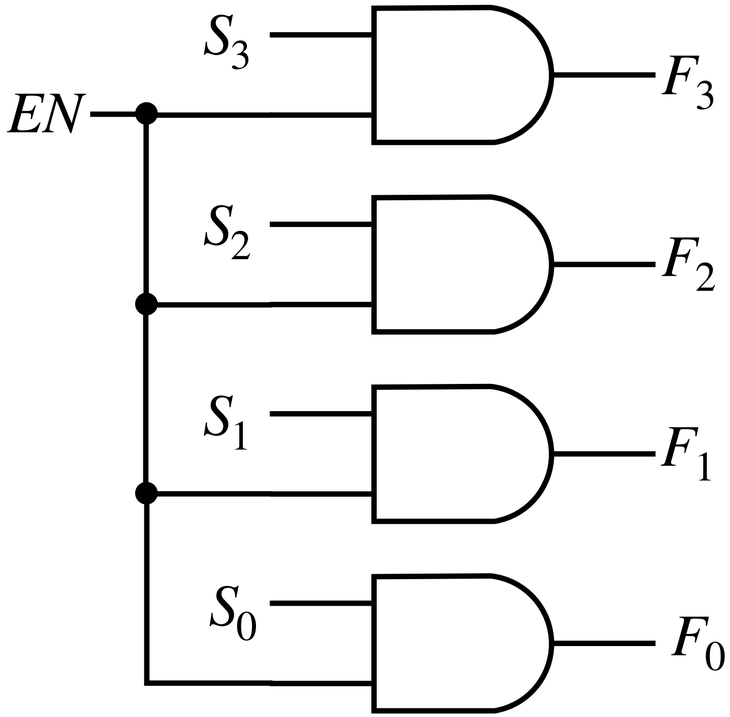
- Transferring function, but with an additional EN signal acting as switch





Enabler

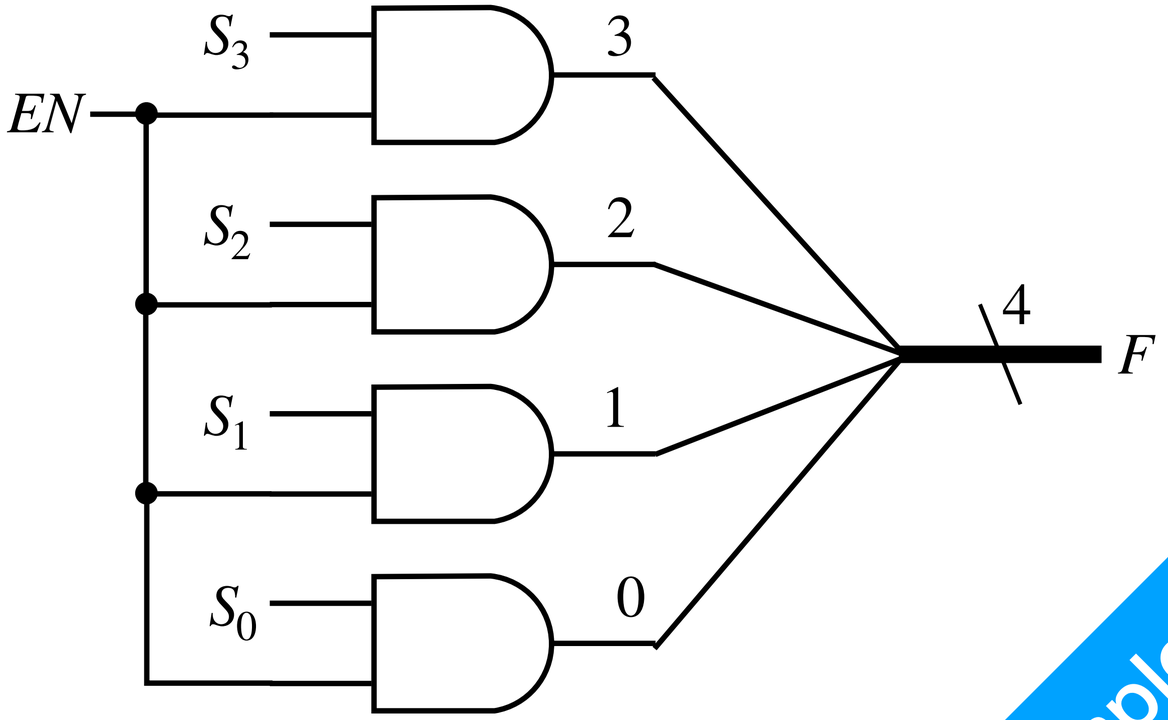
- A building with individual lights $F(3:0)$, and individual switches $S(3:0)$
  - S_i controls F_i
- Master switch: EN



Example

Enabler

- A building with individual lights $F(3:0)$, and individual switches $S(3:0)$
  - S_i controls F_i
- Master switch: EN



④ Multiple-bit Function

Example

Summary

① **Value-Fixing**

② **Transferring**

③ **Inverting**

④ **Multiple-bit Function**

⑤ **Enabler**

Summary

Decoding

n-bit input, 2^n-bit output

Decoder

- n-bit input
  - 2^n different combinations
- Decoder
  - n-bit input, n-2^n output
    each unique input produces a unique output

Concept

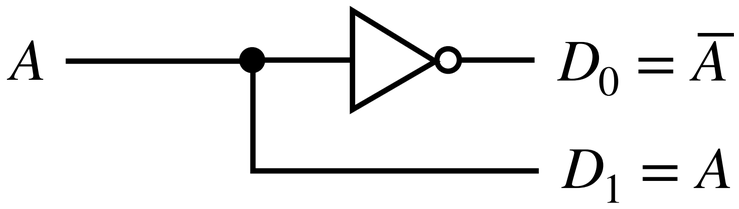
P2
Decoder

1-to-2 Decoder

Technology
- 1 x NOT Gate

- 1bit input, 2bits output

A	D_0	D_1
0	1	0
1	0	1



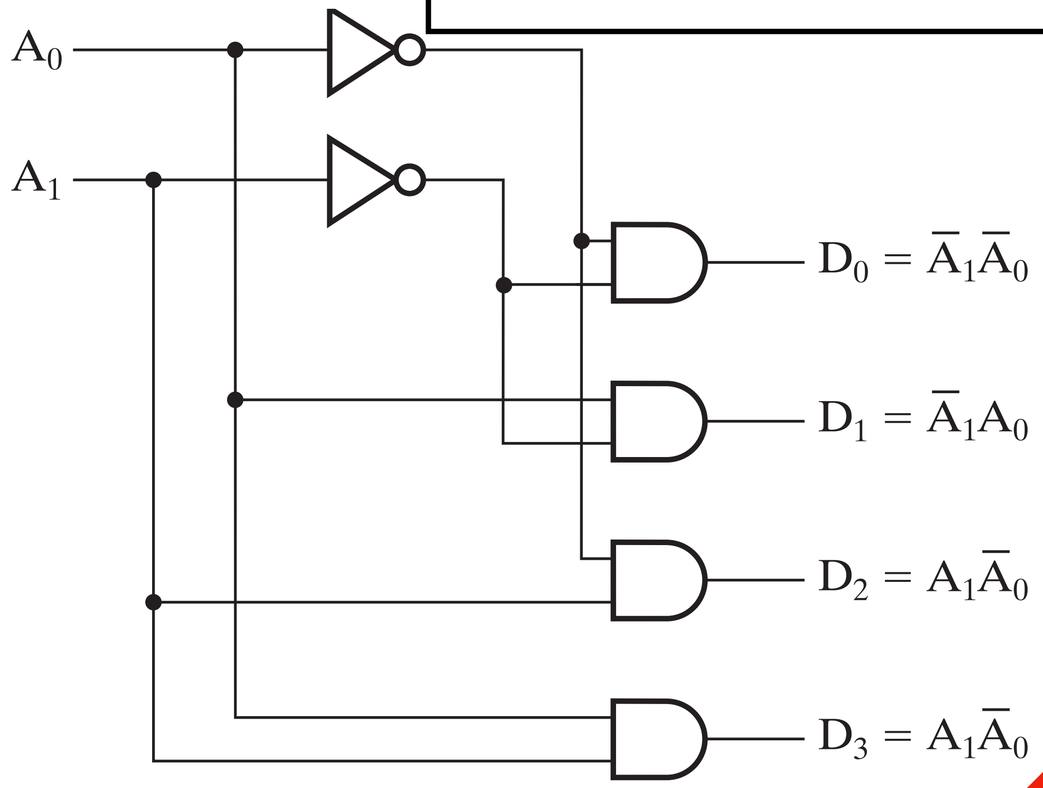
Concept

2-to-4 Decoder

Technology
- 2 x NOT Gate
- 4 x 2-input AND Gate

- 2bit input, 4bits output
  - $D_i = m_i$

A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



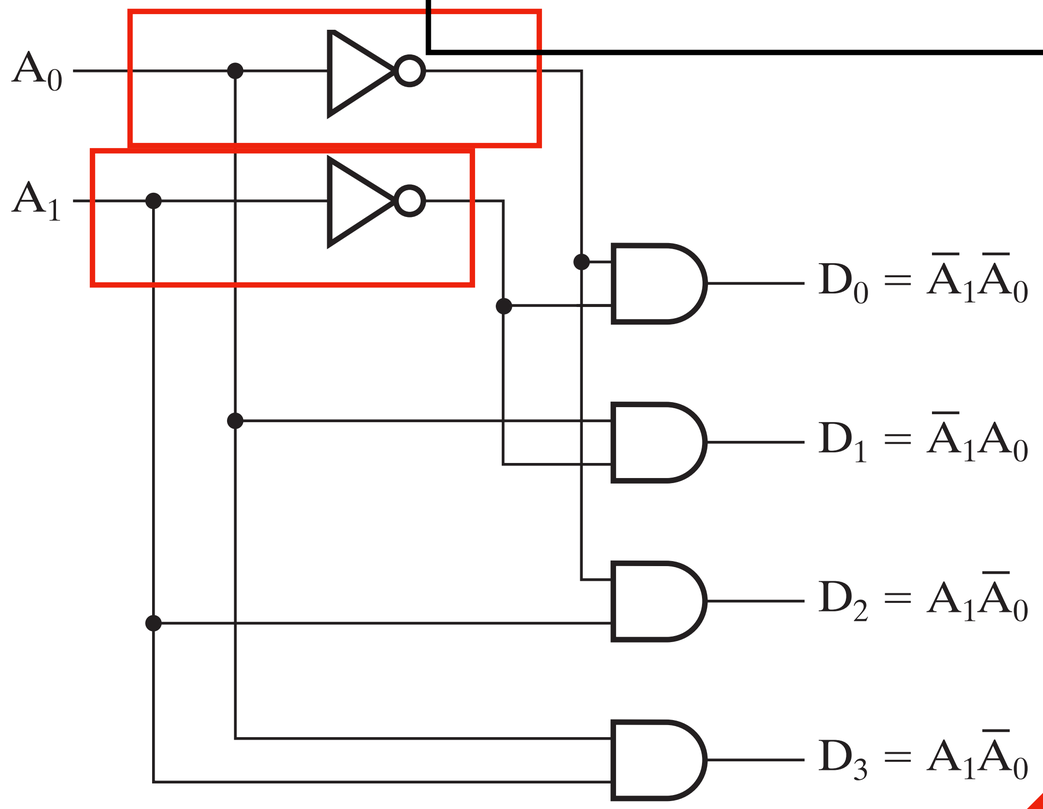
Concept

2-to-4 Decoder

Technology
- 2 x NOT Gate
- 4 x 2-input AND Gate

- 2bit input, 4bits output
  - $D_i = m_i$

A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



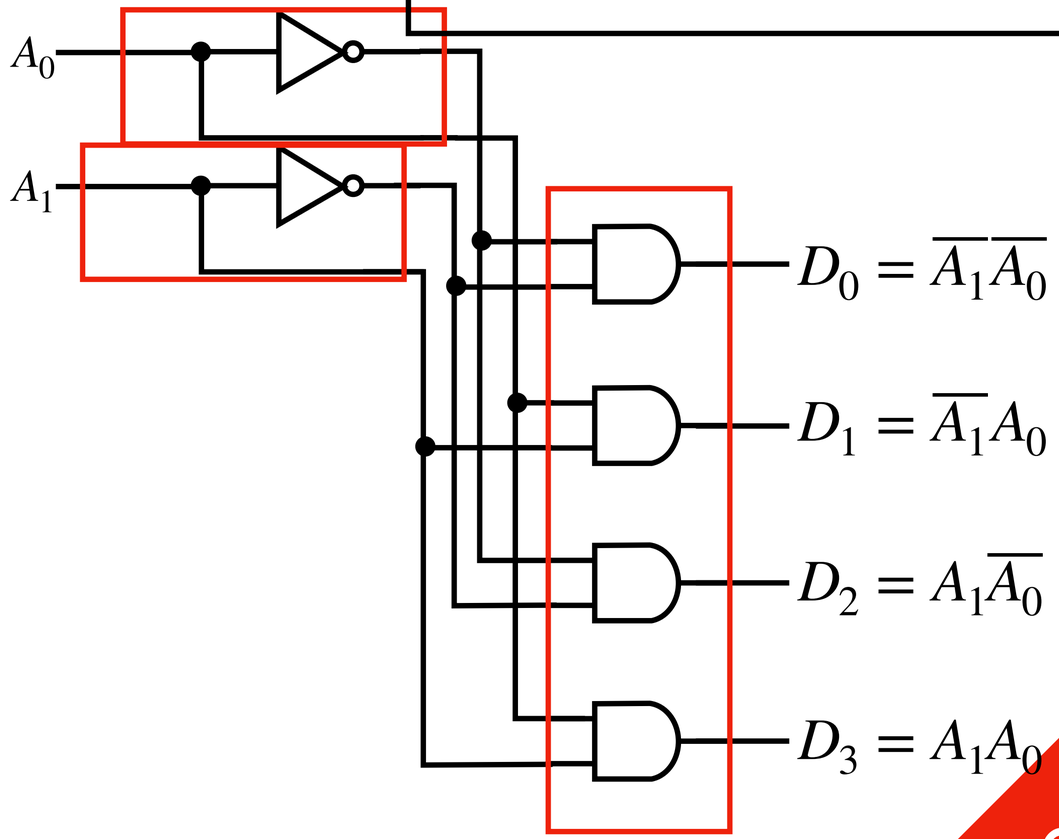
Concept

2-to-4 Decoder

Technology
- 2 x 1-to-2 Decoder
- 4 x 2-input AND Gate

- 2bit input, 4bits output
  - $D_i = m_i$

A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Concept

3-to-8 Decoder

Technology
- 1 x 1-to-2 Decoder
- 1 x 2-to-4 Decoder
- 8 x 2-input AND Gate

- 3bit input, 8bits output
  - $D_i = m_i$

A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
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

Concept

3-to-8 Decoder

Technology
- 1 x 1-to-2 Decoder
- 1 x 2-to-4 Decoder
- 8 x 2-input AND Gate

- 3bit input, 8bits output
  - $D_i = m_i$

A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
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

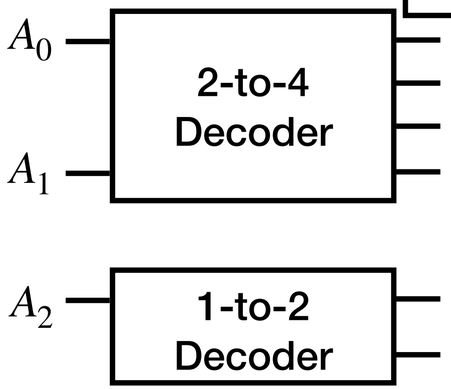
Concept

3-to-8 Decoder

Technology
- 1 x 1-to-2 Decoder
- 1 x 2-to-4 Decoder
- 8 x 2-input AND Gate

- 3bit input, 8bits output
  - $D_i = m_i$

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



$$D_0^3 = \overline{A}_2 D_0^2$$
$$D_1^3 = \overline{A}_2 D_1^2$$
$$D_2^3 = \overline{A}_2 D_2^2$$
$$D_3^3 = \overline{A}_2 D_3^2$$
$$D_4^3 = A_2 D_0^2$$
$$D_5^3 = A_2 D_1^2$$
$$D_6^3 = A_2 D_2^2$$
$$D_7^3 = A_2 D_3^2$$

3-to-8 Decoder

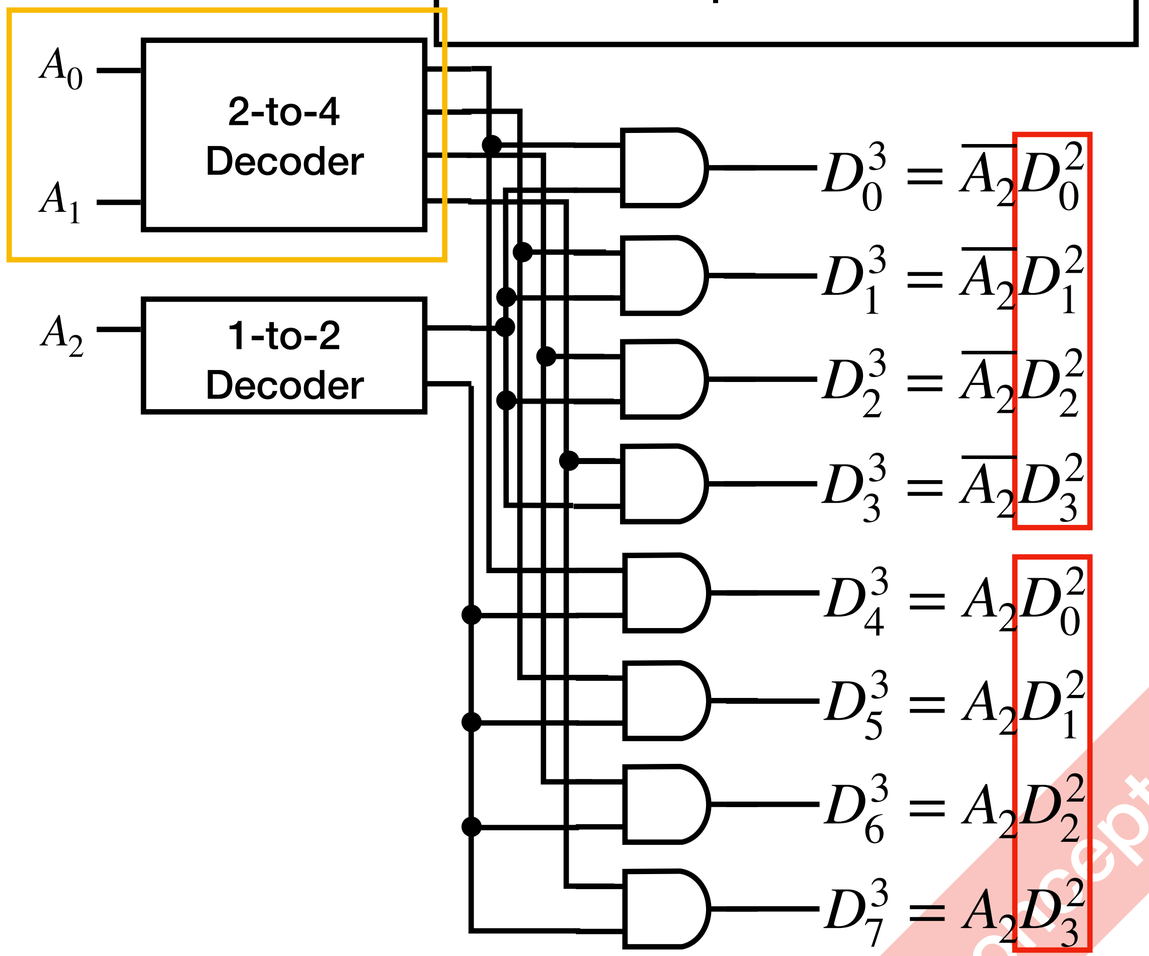
Incremental Design

- 3bit input, 8bits output
  - $D_i = m_i$

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

Technology

- 1 x 1-to-2 Decoder
- 1 x 2-to-4 Decoder
- 8 x 2-input AND Gate



4-to-16 Decoder

Incremental Design

Technology
- 1 x 1-to-2 Decoder
- 1 x 3-to-8 Decoder
- 16 x 2-input AND Gate

- 4bit input, 16bits output
  - $D_i = m_i$

$$D^4_{0:7} = \overline{A_3} D^3_{0:7} \quad D^4_{8:15} = A_3 D^3_{0:7}$$

A_3	A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	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

A_3	A_2	A_1	A_0	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}
1	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

Concept

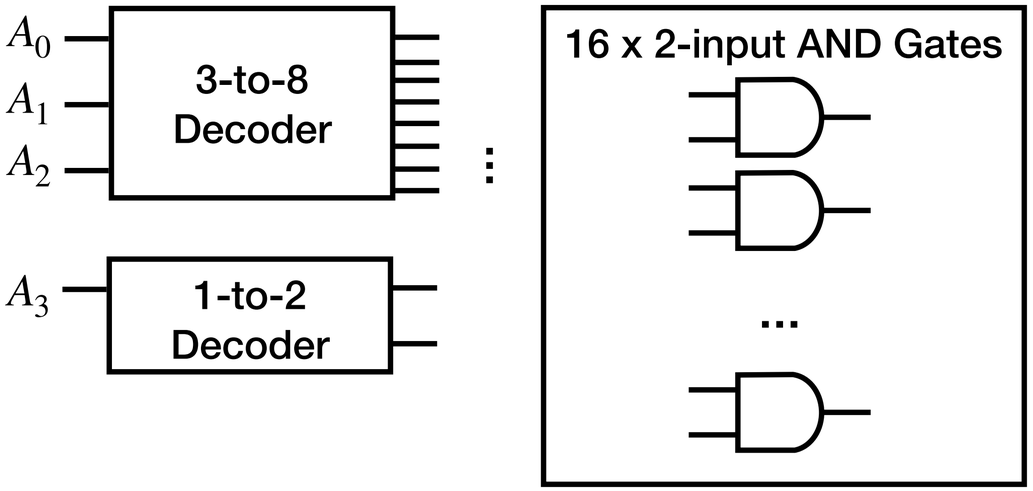
1. Incomplete Truth table

4-to-16 Decoder

Incremental Design

Technology
- 1 x 1-to-2 Decoder
- 1 x 3-to-8 Decoder
- 16 x 2-input AND Gate

- 4bit input, 16bits output
  - $D_i = m_i$



$$D^4_{0:7} = \overline{A_3} D^3_{0:7}$$

$$D^4_{8:15} = A_3 D^3_{0:7}$$

Concept

4-to-16 Decoder

Recursive Design

Technology
- 2 x 2-to-4 Decoder
- 16 x 2-input AND Gate

- 4bit input, 16bits output
  - $D_i = m_i$

$$D^4_{0:7} = \overline{A_3} D^3_{0:7} \quad D^4_{8:15} = A_3 D^3_{0:7}$$

A_3	A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	0	1	0	0	0	0	0	0	0
		0	1	0	1	0	0	0	0	0	0
		1	0	0	0	1	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
		0	1	0	0	0	0	0	1	0	0
		1	0	0	0	0	0	0	0	1	0
		1	1	0	0	0	0	0	0	0	1

A_3	A_2	A_1	A_0	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}
1	0	0	0	1	0	0	0	0	0	0	0
		0	1	0	1	0	0	0	0	0	0
		1	0	0	0	1	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
		0	1	0	0	0	0	0	1	0	0
		1	0	0	0	0	0	0	0	1	0
		1	1	0	0	0	0	0	0	0	1

1. Incomplete Truth table

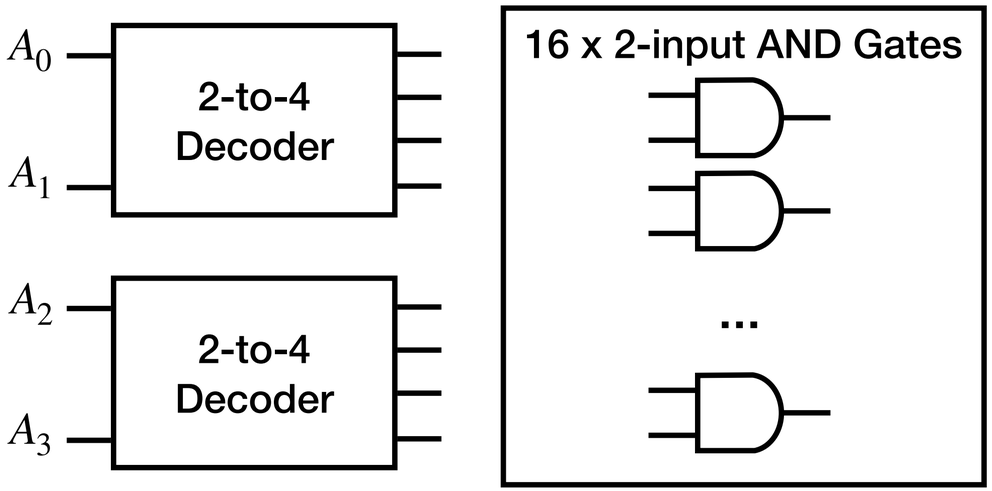
4-to-16 Decoder

Recursive Design

Technology
- 2 x 2-to-4 Decoder
- 16 x 2-input AND Gate

- 4bit input, 16bits output
  - $D_i = m_i$

$$D^4_{0:7} = \overline{A_3} D^3_{0:7} \quad D^4_{8:15} = A_3 D^3_{0:7}$$



$$D^4_{0:3} = \overline{A_3}\,\overline{A_2} D^2$$
$$D^4_{4:7} = \overline{A_3} A_2 D^2$$
$$D^4_{8:11} = A_3 \overline{A_2} D^2$$
$$D^4_{12:15} = A_3 A_2 D^2$$

Concept

1. Incomplete Truth table

P2
Decoder

4-to-16 Decoder

Recursive Design

Technology
- 2 x 2-to-4 Decoder
- 16 x 2-input AND Gate

- 4bit input, 16bits output
  - $D_i = m_i$

$$D^4_{0:7} = \overline{A_3} D^3_{0:7}$$
$$D^4_{8:15} = A_3 D^3_{0:7}$$

Concept

P2
Decoder

4-to-16 Decoder

Recursive Design

Technology
- 2 x 2-to-4 Decoder
- 16 x 2-input AND Gate

- 4bit input, 16bits output
  - $D_i = m_i$

$$D^4 = D^4_{0:15} = [\overline{A_3} D^3_{0:7}; A_3 D^3_{0:7}] = [\overline{A_3} D^3; A_3 D^3]$$

Bracket: concatenation of vectors

Concept

P2
Decoder

4-to-16 Decoder

Recursive Design

Technology
- 2 x 2-to-4 Decoder
- 16 x 2-input AND Gate

- 4bit input, 16bits output
  - $D_i = m_i$

$$D^4 = [\overline{A_3}D^3; A_3D^3]$$

Concept

P2
Decoder

4-to-16 Decoder

Technology
- 2 x 2-to-4 Decoder
- 16 x 2-input AND Gate

Recursive Design

- 4bit input, 16bits output
  - $D_i = m_i$

$$D^4 = [\overline{A_3}D^3; A_3D^3]$$

$$D^3 = \left[\begin{pmatrix}\overline{A_2}D_0^2\\ \overline{A_2}D_1^2\\ \overline{A_2}D_2^2\\ \overline{A_2}D_3^2\end{pmatrix}; \begin{pmatrix}A_2D_0^2\\ A_2D_1^2\\ A_2D_2^2\\ A_2D_3^2\end{pmatrix}\right] = [\overline{A_2}\begin{pmatrix}D_0^2\\ D_1^2\\ D_2^2\\ D_3^2\end{pmatrix}; A_2\begin{pmatrix}D_0^2\\ D_1^2\\ D_2^2\\ D_3^2\end{pmatrix}] = [\overline{A_2}D^2; A_2D^2]$$

Concept

P2
Decoder

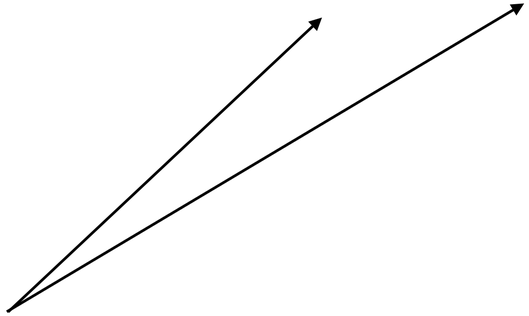
4-to-16 Decoder

Technology
- 2 x 2-to-4 Decoder
- 16 x 2-input AND Gate

Recursive Design

- 4bit input, 16bits output
  - $D_i = m_i$

$$D^4 = [\overline{A_3} D^3; A_3 D^3]$$



$$D^3 = [\overline{A_2} D^2; A_2 D^2]$$

Concept

P2
Decoder

4-to-16 Decoder

Recursive Design

Technology
- 2 x 2-to-4 Decoder
- 16 x 2-input AND Gate

- 4bit input, 16bits output
  - $D_i = m_i$

$$D^3 = [\overline{A_2}D^2; A_2D^2]$$

$$D^4 = [\overline{A_3}D^3; A_3D^3] = [\overline{A_3}[\overline{A_2}D^2; A_2D^2]; A_3[\overline{A_2}D^2; A_2D^2]]$$

$$= [\overline{A_3}\,\overline{A_2}D^2; \overline{A_3}A_2D^2]; [A_3\overline{A_2}D^2; A_3A_2D^2]]$$

$$= [\overline{A_3}\,\overline{A_2}D^2; \overline{A_3}A_2D^2; A_3\overline{A_2}D^2; A_3A_2D^2]$$

$$= [\overline{A_3}\,\overline{A_2}; \overline{A_3}A_2; A_3\overline{A_2}; A_3A_2]D^2$$

Concept

Summary

- What is a decoder
- Truth table of a decoder
- Implementation of 1-to-2, 2-to-4, n-to-2^ndecoder
  - Incremental design
  - Recursive design

P3
Example 3d

1-bit Binary Adder

Using decoder

Summary

P3
Example 3d

Binary Addition

Carries Z	11010
Augend X	01101
Addend Y	+00101
Sum	10010

Review

P3
Example 3d

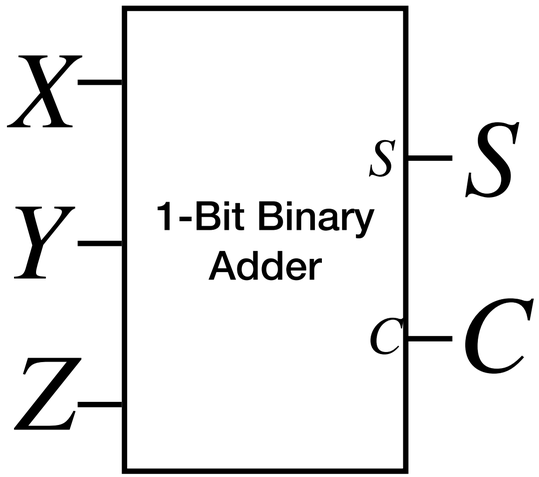
1-bit Binary Adder

- Binary Adder
  - Logical functional block that performs addition
- 1-Bit Binary Adder: smallest building block of a multi-bit adder
  - Inputs
    Augend: X; Addend: Y; Previous Carry: Z
  - Outputs
    Sum bit: S; next carry: C

Concept

P3
Example 3d

1. Specification



Carries $C\,Z$

Input
Output

Augend X

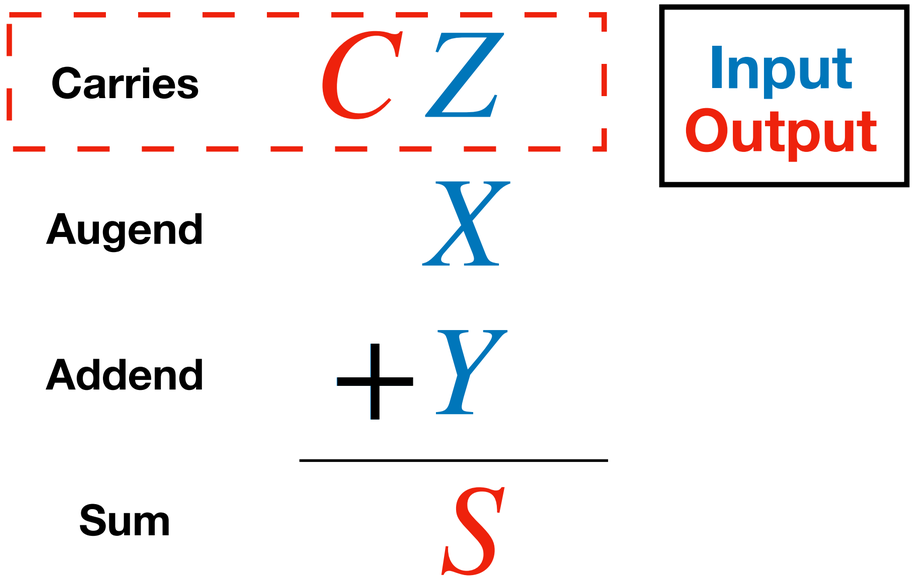
Addend $+\,Y$

Sum S

Example

P3
Example 3d

2. Formulation



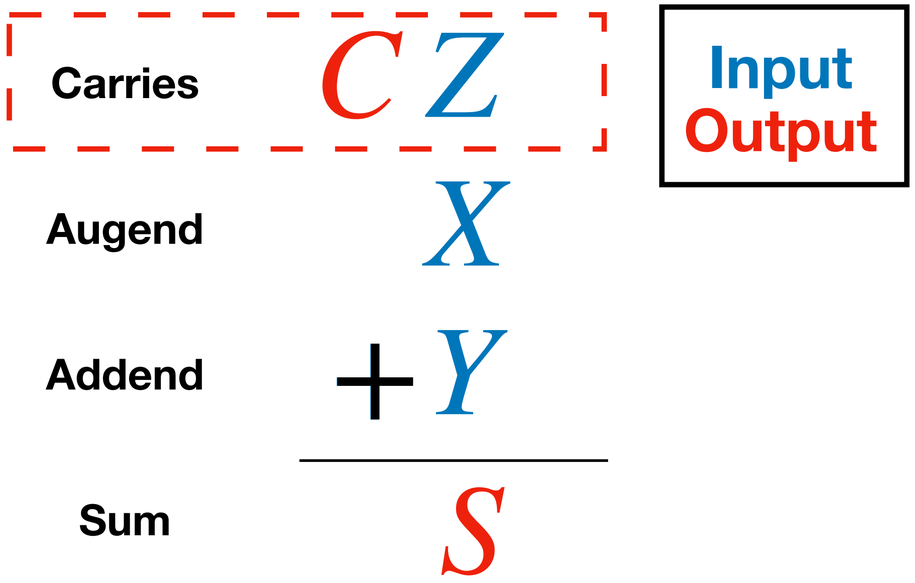
Input
Output

X	Y	Z	S	C
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Example

P3
Example 3d

2. Formulation



Input
Output

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Example

P3
Example 3d

2. Formulation

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sum of Minterms

$$C = \Sigma m(3,5,6,7)$$

$$S = \Sigma m(1,2,4,7)$$

Example

P3
Example 3d

3. Optimisation

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sum of Minterms

$$C = \Sigma m(3,5,6,7)$$

$$S = \Sigma m(1,2,4,7)$$

We can use decoders to implement the adder!

Example

P3
Example 3d

3. Optimisation

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sum of Minterms

$$C = \Sigma m(3,5,6,7)$$

$$S = \Sigma m(1,2,4,7)$$

We can use decoders to implement the adder!

Example

P3
Example 3d

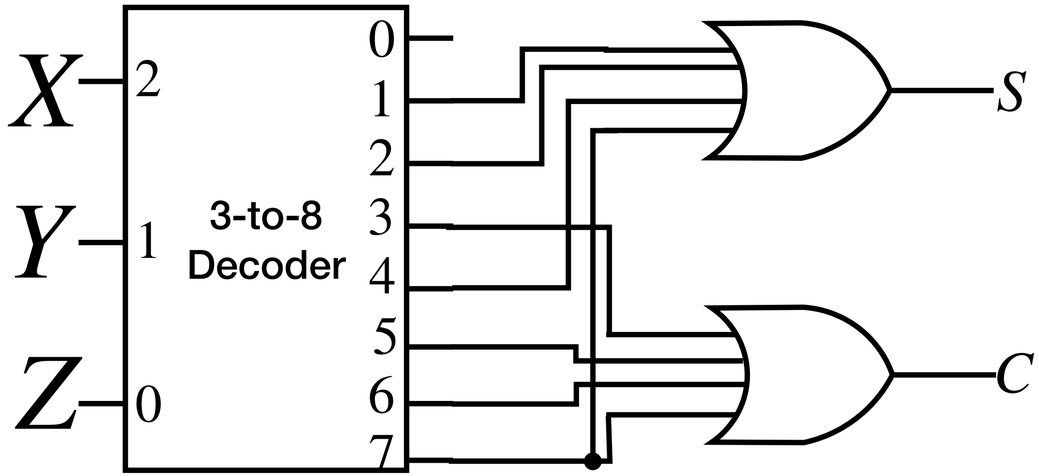
4. Technology Mapping

Technology
- 1 x 3-to-8 Decoder
- OR Gates

Sum of Minterms

$$C = \Sigma m(3,5,6,7)$$

$$S = \Sigma m(1,2,4,7)$$



Example