Combinational Systems

course	
area	ucla
created	@January 12, 2023 2:11 PM
updated	@January 13, 2023 10:30 AM
✅ tasks	
📕 courses	
URL	
tags	cs

Definitions

- e.g. combinational system

  Input: $x \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  Output: $z \in \{0, 1, 2\}$

  Function: F is described by the following table

x	0	1	2	3	4	5	6	7	8	9
$z = F(x)$	0	1	2	0	1	2	0	1	2	0

  or by the arithmetic expression

  $$z = x \bmod 3,$$

  - specify binary conversion “helper functions”

Resources

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a94d2302-b0a8-4841-9bb8-5577bd427e32/ch2.pdf

$z = x \bmod 3,$

x	0	1	2	3	4	5	6	7	8	9
$\underline{x}_b = C(x)$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

$\underline{z}_b$	00	01	10
$z = D(\underline{z}_b)$	0	1	2

- then re-implement I/O sets to define the function

Input: $\underline{x}_b = (x_3, x_2, x_1, x_0),\ x_i \in \{0, 1\}$
Output: $\underline{z}_b = (z_1, z_0),\ z_i \in \{0, 1\}$

Function: F_b is described by the following table

$\underline{x}_b$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
$\underline{z}_b = F_b(\underline{x}_b)$	00	01	10	00	01	10	00	01	10	00

- e.g. conditional expression (input-output func)

Inputs: $\underline{x} = (x_3, x_2, x_1, x_0),$
$x_i \in \{\mathrm{A}, \mathrm{B}, \ldots, \mathrm{Z}, \mathrm{a}, \mathrm{b}, \ldots, \mathrm{z}\}$
$y \in \{\mathrm{A}, \mathrm{B}, \ldots, \mathrm{Z}, \mathrm{a}, \mathrm{b}, \ldots, \mathrm{z}\}$
$k \in \{0, 1, 2, 3\}$
Outputs: $\underline{z} = (z_3, z_2, z_1, z_0),$
$z_i \in \{\mathrm{A}, \mathrm{B}, \ldots, \mathrm{Z}, \mathrm{a}, \mathrm{b}, \ldots, \mathrm{z}\}$

Function: $z_j = \begin{cases} x_j & \textbf{if} \quad j \neq k \\ y & \textbf{if} \quad j = k \end{cases}$

Input: $\underline{x} = (\mathrm{C{,}A{,}S{,}E})$, $y = \mathrm{R}$, $k = 1$
Output: $\underline{z} = (\mathrm{C{,}A{,}R{,}E})$

- e.g. boolean algebra

$$
\begin{aligned}
E_1(x_2, x_1, x_0) &= x_2 x_1 + x_2 x_1' + x_2 x_0 \\
E_2(x_2, x_1, x_0) &= x_2
\end{aligned}
$$

$$
\begin{aligned}
x_2 x_1 + x_2 x_1' + x_2 x_0 &= x_2(x_1 + x_1') + x_2 x_0 && \text{using } ab + ac = a(b + c) \\
&= x_2 \cdot 1 + x_2 x_0 && \text{using } a + a' = 1 \\
&= x_2(1 + x_0) && \text{using } ab + ac = a(b + c) \\
&= x_2 \cdot 1 && \text{using } 1 + a = 1 \\
&= x_2 && \text{using } a \cdot 1 = a
\end{aligned}
$$

- e.g. base 2^n conversion

(B2451)_16 : 010 110 010 010 001 010 001
(x)_8 : 2 6 2 2 1 2 1
-> x = (2622121)_8

Big Ideas

▼ Input-Output Functions

- ▼ described by the function $z(t) = F(x(t)) \implies z = F(x)$

▼ Binary Level

- ▼ $z_b = F_b(x_b)$
- ▼ e.g. a high level overview

$z_b = F_b(x_b)$

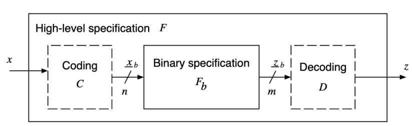


Figure 2.2: High-level and binary-level specification.

▼ High Level Spec

- ▼ input set - set of input values
- ▼ output set - set of output values
- ▼ input-output function - $z = F(x)$
    - ▼ can be specified as:
    - ▼ table

x	z
A	65
B	66
C	67
D	68
E	69

- arithmetic

$$z = 3x + 2y - 2$$

- conditional

$$z = \begin{cases} a + b & \textbf{if} \quad c > d \\ a - b & \textbf{if} \quad c = d \\ 0 & \textbf{if} \quad c < d \end{cases}$$

- logical

$$\begin{aligned} z = {} & (\textsc{switch1} = \textsc{closed}) \textbf{ and } (\textsc{switch2} = \textsc{open}) \\ & \textbf{or } (\textsc{switch3} = \textsc{closed}) \end{aligned}$$

- composition of function

$$\textsc{max}(v, w, x, y) = \textsc{greater}(v, \textsc{greater}(w, \textsc{greater}(x, y)))$$

in which

$$\textsc{greater}(a, b) = \begin{cases} a & \textbf{if} \quad a > b \\ b & \textbf{otherwise} \end{cases}$$

▼ Encoding

▼ ASCII

▼ a standardized bit-vector encoding for character which include letters, digits, and special characters

▼ Integer Encoding

▼ integer ←→ digit-vector

▼ digit ←→ bit-vector

▼ e.g. 4-digit integer

Level 1: Integer (Digit-vector)	5				6				3				0			
Level 2: Bit-vector	1	0	1	1	1	0	0	1	1	0	0	0				

▼ Binary Encoding Methods

▼ Gray Code

▼ a form of binary encoding that implements character conversion <u>differently</u>

▼ gray code changes only a singular bit between two adjacent characters (whereas binary encoding can change multiple)

▼ e.g. gray code example

15

Binary-code representation of a digit-vector

<table>
<tr><td colspan="3">5</td><td colspan="3">6</td><td colspan="3">3</td><td colspan="3">0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
</table>

a) Gray code. b) Gray-code bit-vector representation of a digit-vector.

Digit	Gray code
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

(a)

<table>
<tr><td colspan="3">4</td><td colspan="3">5</td><td colspan="3">3</td><td colspan="3">0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

(b)

▼ Other Binary Encoding Method

Digit Value	BCD 8421	2421	Excess-3	2-Out-of-5
0	0000	0000	0011	00011
1	0001	0001	0100	11000
2	0010	0010	0101	10100
3	0011	0011	0110	01100
4	0100	0100	0111	10010
5	0101	1011	1000	01010
6	0110	1100	1001	00110
7	0111	1101	1010	10001
8	1000	1110	1011	01001
9	1001	1111	1100	00101

▼ General Encoding Principle

- ▼ radix sum method

$$(x)_{10} = \sum_{i}^{n-1} x_i r^i$$

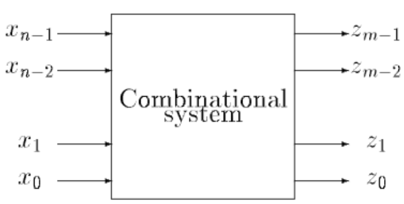
- ▼ base 2^n encoding conversion
  - ▼ $(x)_{2^n} = (y)_{2^r}$
  - ▼ $\log_2 2^n = n$

- ▼ n is the number of binary digits you should have
- ▼ r is the number of binary digit subsets of n
- ▼ prepend 0 to follow base conversion

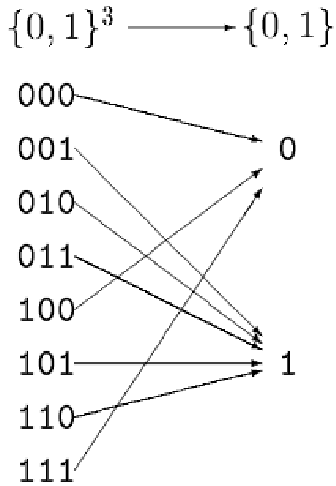
▼ Switching Functions

▼ Comb. Sys. vs. Switching Functions

- ▼ e.g. binary combinational system



- ▼ representing the previous comb. sys. as a switching function for n bit-encoding



(b)

▼ Tabular Representation

n-tuple notation		Simplified notation	
$x_2x_1x_0$	$f(x_2,x_1,x_0)$	j	$f(j)$
0 0 0	0	0	0
0 0 1	0	1	0
0 1 0	1	2	1
0 1 1	1	3	1
1 0 0	0	4	0
1 0 1	0	5	0
1 1 0	1	6	1
1 1 1	1	7	1

	$x_2x_1x_0$							
x_4x_3	000	001	010	011	100	101	110	111
00	0	0	1	1	0	1	1	1
01	0	1	1	1	1	0	1	1
10	1	1	0	1	1	0	1	1
11	0	1	0	1	1	0	1	0

f

▼ Special Switching Functions

Table 2.10: Switching functions of one variable

x	f_0 0-CONSTANT (always 0)	f_1 IDENTITY (equal to x)	f_2 COMPLEMENT (NOT)	f_3 1-CONSTANT (always 1)
0	0	0	1	1
1	0	1	0	1

Table 2.11: Switching functions of two variables

Function	x_1x_0 00	01	10	11	
f_0	0	0	0	0	
f_1	0	0	0	1	AND
f_2	0	0	1	0	
f_3	0	0	1	1	
f_4	0	1	0	0	
f_5	0	1	0	1	
f_6	0	1	1	0	EXCLUSIVE-OR (XOR)
f_7	0	1	1	1	OR
f_8	1	0	0	0	NOR
f_9	1	0	0	1	EQUIVALENCE (EQU)
f_{10}	1	0	1	0	
f_{11}	1	0	1	1	
f_{12}	1	1	0	0	
f_{13}	1	1	0	1	
f_{14}	1	1	1	0	NAND
f_{15}	1	1	1	1	

▼ Switching Expressions (Boolean)

- ▼ symbol representing binary/boolean variables are SEs

▼ SE Arithmetic (of A, B)

- ▼ (A)' is a SE - the "A complement"
- ▼ $(A) + (B)$ is a SE - "A or B"
- ▼ $(A) \cdot (B)$ is a SE - "A and B"
- ▼ order of operations:
  - ▼ complement → prod → sum

▼ SE (Boolean) Algebra

- *Switching algebra*:

  two elements 0 and 1
  operations $+$, $\cdot$, and $'$

$+$	0	1
0	0	1
1	1	1

$\cdot$	0	1
0	0	0
1	0	1

$'$	
0	1
1	0

$$E(x_2, x_1, x_0) = x_2 + x_2'x_1 + x_1x_0'$$

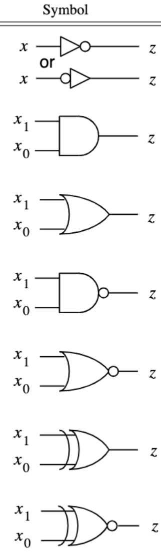
The value of E for assignment $(1, 0, 1)$ is

$$E(1, 0, 1) = 1 + 1' \cdot 0 + 0 \cdot 1' = 1 + 0 + 0 = 1$$

▼ Algebraic Properties

1.	$a + b$	$= b + a$	ab	$= ba$	Commutativity
2.	$a + (bc)$	$= (a + b)(a + c)$	$a(b + c)$	$= (ab) + (ac)$	Distributivity
3.	$a + (b + c)$	$= (a + b) + c$	$a(bc)$	$= (ab)c$	Associativity
		$= a + b + c$		$= abc$	
4.	$a + a$	$= a$	aa	$= a$	Idempotency
5.	$a + a'$	$= 1$	aa'	$= 0$	Complement
6.	$1 + a$	$= 1$	$0a$	$= 0$	
7.	$0 + a$	$= a$	$1a$	$= a$	Identity
8.	$(a')'$	$= a$			Involution
9.	$a + ab$	$= a$	$a(a + b)$	$= a$	Absorption
10.	$a + a'b$	$= a + b$	$a(a' + b)$	$= ab$	Simplification
11.	$(a + b)'$	$= a'b'$	$(ab)'$	$= a' + b'$	DeMorgan's Law

▼ 2-var Logic Gates as SE



Gate type	Symbol	Switching expression
NOT	$x \to z$ or $x \to z$	$z = x'$
AND	$x_1, x_0 \to z$	$z = x_1x_0$
OR	$x_1, x_0 \to z$	$z = x_1 + x_0$
NAND	$x_1, x_0 \to z$	$z = (x_1x_0)'$
NOR	$x_1, x_0 \to z$	$z = (x_1 + x_0)'$
XOR	$x_1, x_0 \to z$	$z = x_1x_0' + x_1'x_0 = x_1 \oplus x_0$
XNOR	$x_1, x_0 \to z$	$z = x_1'x_0' + x_1x_0$

▼ N-var Logic Gates as SE

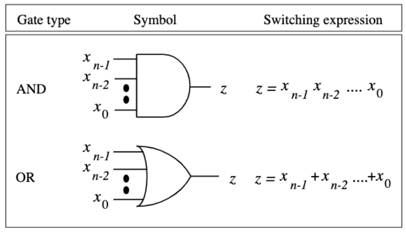


Figure 2.5: n-input AND and OR gate symbols

▼ Equivalence Classes representation

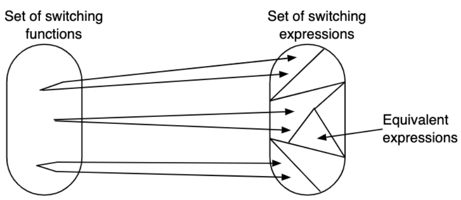
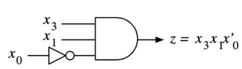


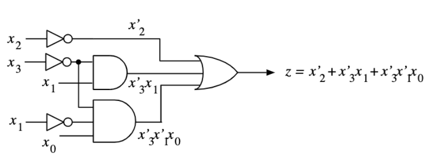
Figure 2.6: Correspondence among switching functions and switching expressions

▼ Logic Gate Expression

Literals	$x,\ y,\ z',\ x'$
Product terms	$x_0,\ x_2x_1,\ x_3x_1x_0'$
Sum of products	$x_2' + x_3x_1' + x_3'x_1'x_0$



(a)



(b)

▼ Minterm/Maxterm Notation

▼ representing SEs as a simplified notation by defining a type conversion

- minterm - sum of products
- maxterm - product of sums
- e.g. minterms

$$x_i \longleftrightarrow 1; \qquad x_i' \longleftrightarrow 0$$

Minterm m_j, j integer

Example: minterm $x_3 x_2' x_1' x_0$ denoted m_9
because $1001 = 9$

$$m_j(\underline{a}) = \begin{cases} 1 & \textbf{if} \quad a = j \\ 0 & \textbf{otherwise} \end{cases}$$

$$a = \sum_{i=0}^{n-1} a_i 2^i$$

Example: $m_{11} = x_3 x_2' x_1 x_0$
– has value 1 only for $\underline{a} = (1,0,1,1)$

Tabular Representation

$x_2x_1x_0$	m_0 $x_2'x_1'x_0'$	m_1 $x_2'x_1'x_0$	m_2 $x_2'x_1x_0'$	m_3 $x_2'x_1x_0$	m_4 $x_2x_1'x_0'$	m_5 $x_2x_1'x_0$	m_6 $x_2x_1x_0'$	m_7 $x_2x_1x_0$
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

Logic Gate Representation

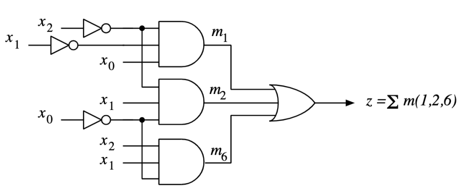


Figure 2.8: Gate network corresponding to $E(x_2, x_1, x_0) = \sum m(1,2,6)$.

📌 **SUMMARY**