

For small digital circuits, the design approaches presented in Chapters 2 and 3 are very effective. However, it is usually impractical to describe a large-scale circuit with a single truth table, minterm list, or set of logic equations. Structured top-down design methodologies must be utilized to effectively manage the complexity of large designs. In this chapter we introduce top-down modular design methods for combinational logic circuits. Then we will examine a number of common combinational logic modules. For each module type we will study its basic function, gate-level circuit realizations of the function, and how the module is used to create larger circuits. The top-down modular design process will then be illustrated by means of a comprehensive example in which a computer arithmetic/logic unit will be designed. The chapter will conclude with a discussion of computer-aided design support for modular design.

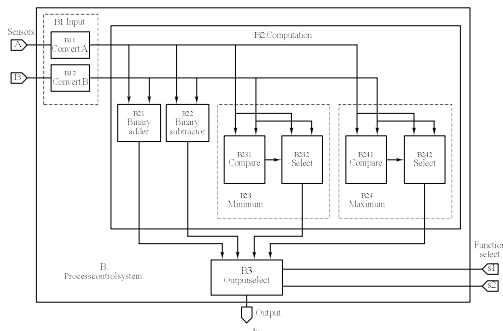
1

```

graph TD
    B1[B Data acquisition system] --> B2_1[B Input sensor data]
    B1 --> B2_2[B Compute values]
    B1 --> B2_3[B Select output]
    
    B2_1 --> B3_1[B1 Sensor A]
    B2_1 --> B3_2[B12 Sensor B]
    B2_1 --> B3_3[B13 A+B]
    
    B2_2 --> B3_4[B14 A-B]
    B2_2 --> B3_5[B21 Main(A,B)]
    
    B2_3 --> B3_6[B24 Max(A,B)]
    B3_6 --> B3_7[B211 Compute A&B]
    B3_6 --> B3_8[B212 Select Min]
    B3_6 --> B3_9[B241 Compute A&B]
    B3_6 --> B3_10[B242 Select Max]
    
    classDef leaf fill:#fff,stroke:#333,stroke-width:1px;
    classDef node fill:#fff,stroke:#333,stroke-width:1px;
    
    class B1,B2_1,B2_2,B2_3 node;
    class B3_1,B3_2,B3_3,B3_4,B3_5,B3_6,B3_7,B3_8,B3_9,B3_10 leaf;
    
    B3_1 --> star1[*]
    B3_2 --> star2[*]
    B3_3 --> star3[*]
    B3_4 --> star4[*]
    B3_5 --> star5[*]
    B3_6 --> star6[*]
    B3_7 --> star7[*]
    B3_8 --> star8[*]
    B3_9 --> star9[*]
    B3_10 --> star10[*]
  
```

* = Leaf node

2



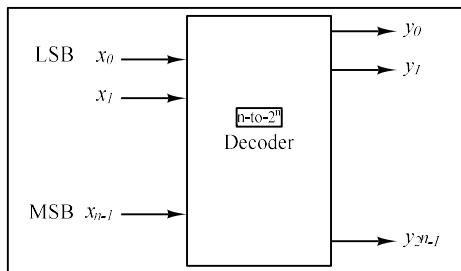
3

Dec	Hx	Oct	Char	Dec	Hx	Oct	Hxmi	Char	Dec	Hx	Oct	Hxmi	Char	Dec	Hx	Oct	Hxmi	Char
0	0	000	(null)	32	20	040	032	Space	64	40	100	064	8	96	60	140	096	8
1	0	001	SOH (start of heading)	33	21	041	033	!	65	41	101	065	A	97	61	141	097	A
2	0	002	STX (start of text)	34	22	042	034	2	66	42	102	066	B	98	62	142	098	B
3	0	003	END (end of text)	35	23	043	035	3	67	43	103	067	C	99	63	143	099	C
4	0	004	ETX (end of transmission)	36	24	044	036	4	68	44	104	068	D	100	64	144	100	D
5	0	005	ENQ (enquiry)	37	25	045	037	5	69	45	105	069	E	101	65	145	101	E
6	0	006	ACK (acknowledge)	38	26	046	038	6	70	46	106	070	F	102	66	146	102	F
7	0	007	BELL	39	27	047	039	7	71	47	107	071	G	103	67	147	103	G
8	0	008	BS (backspace)	40	28	050	040	0	72	48	110	072	H	104	68	150	104	H
9	0	009	TAB (horizontal tab)	41	29	051	041	1	73	49	111	073	I	105	69	151	105	I
10	0	010	VT (vertical tab, new line)	42	30	052	042	2	74	50	112	074	J	106	70	152	106	J
11	0	011	VT (vertical tab)	43	31	053	043	3	75	51	113	075	K	107	71	153	107	K
12	0	012	FF (form feed, new page)	44	32	054	044	4	76	52	114	076	L	108	72	154	108	L
13	0	013	CR (carriage return)	45	33	055	045	5	77	53	115	077	M	109	73	155	109	M
14	0	014	SHO (shift out)	46	34	056	046	6	78	54	116	078	N	110	74	156	110	N
15	0	015	SHI (shift in)	47	35	057	047	7	79	55	117	079	O	111	75	157	111	O
16	0	016	DLE (data link escape)	48	36	058	048	8	80	56	118	080	P	112	76	158	112	P
17	0	017	DC1 (device control 1)	49	37	059	049	9	81	57	119	081	Q	113	77	159	113	Q
18	0	018	DC2 (device control 2)	50	38	060	050	0	82	58	120	082	R	114	78	160	114	R
19	0	019	DC3 (device control 3)	51	39	061	051	1	83	59	121	083	S	115	79	161	115	S
20	0	020	DC4 (device control 4)	52	40	062	052	2	84	60	122	084	T	116	80	162	116	T
21	1	021	NAK (negative acknowledge)	53	41	063	053	3	85	61	123	085	U	117	81	163	117	U
22	1	022	SYN (synchronous idle)	54	42	064	054	4	86	62	124	086	V	118	82	164	118	V
23	1	023	ETB (end of trans. block)	55	43	065	055	5	87	63	125	087	W	119	83	165	119	W
24	1	024	CAN (cancel)	56	44	066	056	6	88	64	126	088	X	120	84	166	120	X
25	1	030	EM (end of medium)	57	39	071	057	9	89	59	131	089	Y	121	79	171	121	Y
26	1	032	FS (file separator)	58	34	072	058	0	90	54	132	090	Z	122	74	172	122	Z
27	1	034	RS (record separator)	59	36	073	059	2	91	56	133	091	[	123	76	173	123	[
28	1	036	SB (substitute)	60	30	074	060	<	92	50	134	092	^	124	70	174	124	^
29	1	038	OS (operator separator)	61	30	075	061	9	93	50	135	093	_	125	70	175	125	_
30	1	039	US (unit separator)	62	31	076	062	0	94	51	136	094	0	126	71	176	126	0
31	1	037	037 (unit separator)	63	37	077	063	9	95	5F	137	095	DEL	127	7F	177	127	DEL

4

Decoders

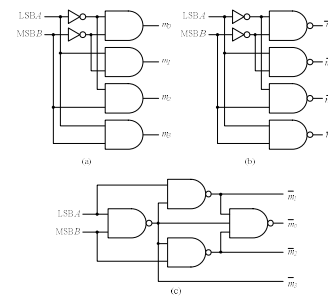
$n - to - 2^n$ decoder module



ASCII, BCD, Gray, Excess-3

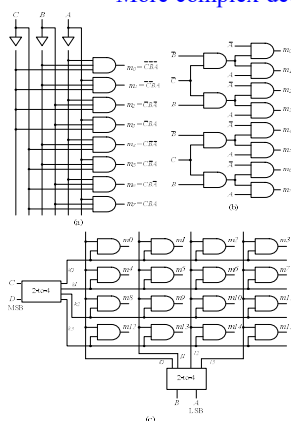
5

Decoder Realization



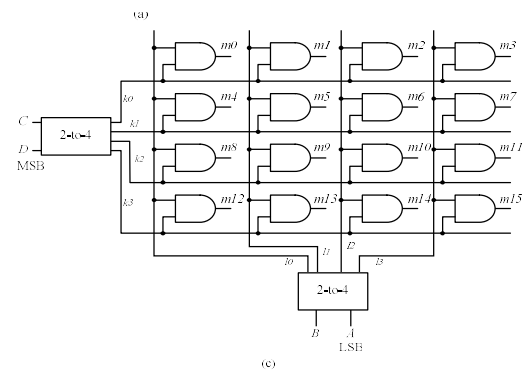
6

More complex decoders

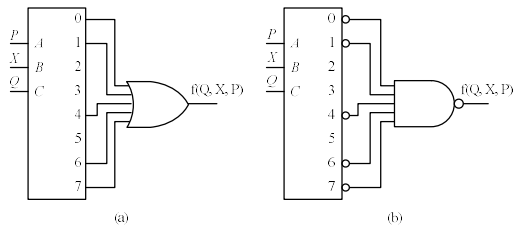


7

More complex decoders

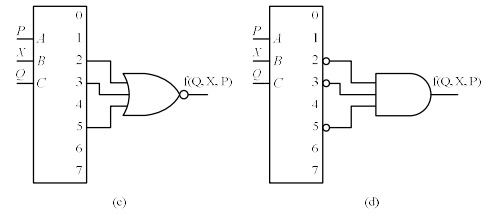


Example 4.1 -- Realize
 $f(Q,X,P) = \sum m(0,1,4,6,7) = \prod M(2,3,5)$



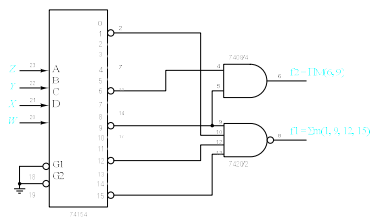
9

Example 4.1 (concluded)



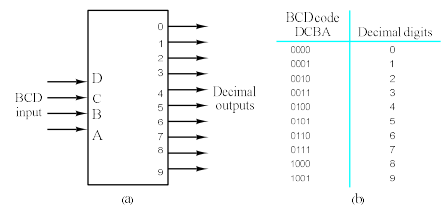
10

Realization of switching functions with a decoder



11

BCD to decimal decoder.

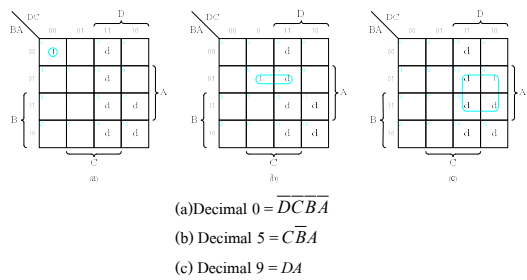


(a) Logic symbol.

(b) BCD codes and decimal digits

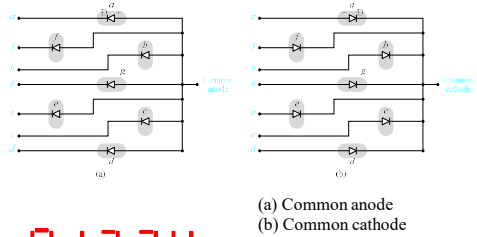
12

K-maps for outputs 0, 5 and 9 of a BCD to decimal code converter



13

7-Segment display elements

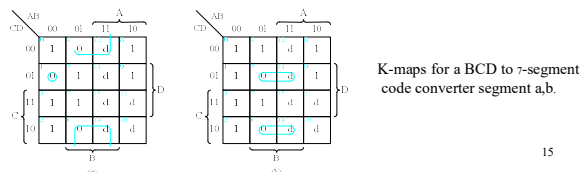


0 1 2 3 4
 5 6 7 8 9 A b C d E F

Decimal digits displayed on 7-segment display element

14

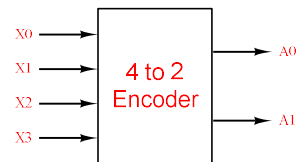
Decimal	BCD Code				Display Segment						
Digit	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
5	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1



15

Encoders

$$2^s \geq n \text{ หรือ } s \geq \log_2 n$$



16

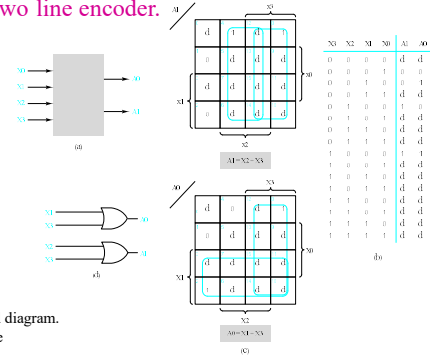
(b) Truth table

X3	X2	X1	X0	A1	A0
0	0	0	0	d	d
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	d	d
0	1	0	0	1	0
0	1	0	1	d	d
0	1	1	0	d	d
0	1	1	1	d	d
1	0	0	0	1	1
1	0	0	1	d	d
1	0	1	0	d	d
1	0	1	1	d	d
1	1	0	0	d	d
1	1	0	1	d	d
1	1	1	0	d	d
1	1	1	1	d	d

(b)

17

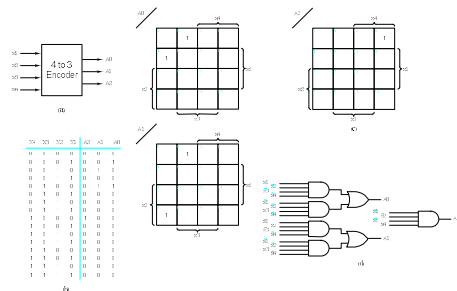
Four-to-two line encoder.



- (a) Functional diagram.
- (b) Truth table
- (c) K-maps
- (d) Logic diagram

18

Four-to-three line encoder



- (a) Functional diagram
- (b) Truth table
- (c) K-maps
- (d) Logic diagram

19

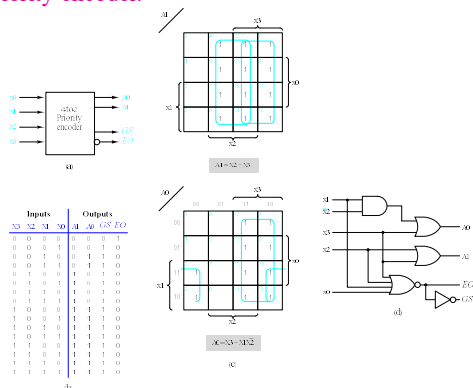
Truth table

x_3	x_2	x_1	x_0	a_2	a_1	a_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	1	0
0	1	0	1	0	1	1
0	1	1	0	1	0	0
0	1	1	1	1	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	1	0
1	0	1	1	1	1	1
1	1	0	0	1	1	0
1	1	0	1	1	1	1
1	1	1	0	1	1	0
1	1	1	1	1	1	1

(b)

20

Priority encoder.

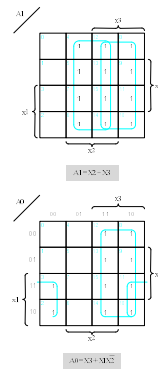


21

(b) Truth table for the 4-to-2 priority encoder.

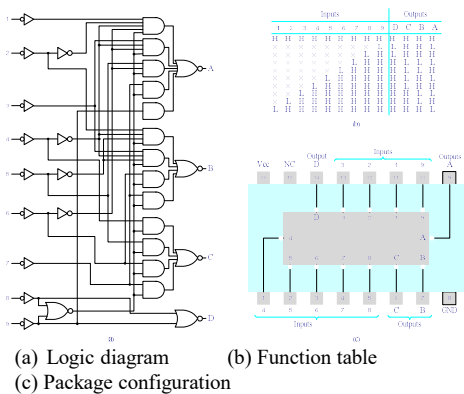
Inputs				Outputs			
x_3	x_2	x_1	x_0	A_1	A_0	GS	EO
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	1	1	0
0	0	1	1	0	1	1	0
0	1	0	0	1	0	1	0
0	1	0	1	1	0	1	0
0	1	1	0	1	1	1	0
0	1	1	1	1	1	1	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	0
1	1	0	0	1	1	1	0
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0

(b)



22

74147 priority encoder module



23

74147 priority encoder module function table

(b) Function table for the 74147 priority encoder module.

Inputs									Outputs			
1	2	3	4	5	6	7	8	9	D	C	B	A
H	H	H	H	H	H	H	H	H	H	H	H	H
X	X	X	X	X	X	X	X	L	L	H	H	L
X	X	X	X	X	X	X	L	H	H	L	H	H
X	X	X	X	X	L	H	H	H	H	L	L	H
X	X	X	X	L	H	H	H	H	H	H	L	H
X	X	X	L	H	H	H	H	H	H	H	L	H
X	X	L	H	H	H	H	H	H	H	H	H	L
X	L	H	H	H	H	H	H	H	H	H	H	L
L	H	H	H	H	H	H	H	H	H	H	H	L

24

74148 priority encoder module

(a) Logic diagram

The logic diagram shows the internal structure of the 74148 priority encoder. It features 8 inputs (I0-I7) and 3 outputs (A2, A1, A0). The inputs are connected to a series of inverters and NAND gates. The outputs are generated by a combination of NAND and OR gates. The diagram illustrates the priority encoding logic, where the highest priority input (I7) is encoded first, and lower priority inputs are only considered if higher priority inputs are inactive.

(b) Function table

Inputs								Outputs			
I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	A_2	A_1	A_0	EO
1	x	x	x	x	x	x	x	1	1	1	0
1	1	x	x	x	x	x	x	1	1	1	0
1	1	1	x	x	x	x	x	1	1	1	0
1	1	1	1	x	x	x	x	1	1	1	0
1	1	1	1	1	x	x	x	1	1	1	0
1	1	1	1	1	1	x	x	1	1	1	0
1	1	1	1	1	1	1	x	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	0
0	1	x	x	x	x	x	x	0	1	1	1
0	1	1	x	x	x	x	x	0	1	1	1
0	1	1	1	x	x	x	x	0	1	1	1
0	1	1	1	1	x	x	x	0	1	1	1
0	1	1	1	1	1	x	x	0	1	1	1
0	1	1	1	1	1	1	x	0	1	1	1
0	1	1	1	1	1	1	1	0	1	1	1
0	0	x	x	x	x	x	x	0	0	0	1
0	0	1	x	x	x	x	x	0	0	0	1
0	0	1	1	x	x	x	x	0	0	0	1
0	0	1	1	1	x	x	x	0	0	0	1
0	0	1	1	1	1	x	x	0	0	0	1
0	0	1	1	1	1	1	x	0	0	0	1
0	0	1	1	1	1	1	1	0	0	0	1
0	0	0	x	x	x	x	x	0	0	0	1
0	0	0	1	x	x	x	x	0	0	0	1
0	0	0	1	1	x	x	x	0	0	0	1
0	0	0	1	1	1	x	x	0	0	0	1
0	0	0	1	1	1	1	x	0	0	0	1
0	0	0	1	1	1	1	1	0	0	0	1
0	0	0	0	x	x	x	x	0	0	0	1
0	0	0	0	1	x	x	x	0	0	0	1
0	0	0	0	1	1	x	x	0	0	0	1
0	0	0	0	1	1	1	x	0	0	0	1
0	0	0	0	1	1	1	1	0	0	0	1
0	0	0	0	0	x	x	x	0	0	0	1
0	0	0	0	0	1	x	x	0	0	0	1
0	0	0	0	0	1	1	x	0	0	0	1
0	0	0	0	0	1	1	1	0	0	0	1
0	0	0	0	0	0	x	x	0	0	0	1
0	0	0	0	0	0	1	x	0	0	0	1
0	0	0	0	0	0	1	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	1

(c) Package configuration

The package configuration shows the pinout for the 74148 priority encoder module. It is a 16-pin DIP package. The inputs are pins 1 through 8 (I0-I7). The outputs are pins 9 through 11 (A2, A1, A0). The enable pins are pins 12 (EO), 13 (EI), and 14 (A2). The output pin 15 is labeled 'A2' and pin 16 is labeled 'A1'.

25

[illegible][illegible]

K-Channel multiplexing/demultiplexing

Diagram illustrating K-Channel multiplexing/demultiplexing:

Inputs: Input 0, Input 1, Input 2, Input 3.

Output: Output.

51 50 (Select pins)

Diagram illustrating IP/ISDN Network Multiplexer:

Countless Possibilities

Diagram showing a network topology with a central IP/ISDN Network Multiplexer connecting various devices (VoIP Router, ISDN Router, IP Network, ISDN Network) and a central IP/ISDN Network Provider.

1 Programmable number of 8-Channels (e.g. 12)

Diagram illustrating Simplex Multiplexing:

Inputs: 4 Cameras (labeled 1 Cable to Camera).

Output: 1 Cable to Monitor.

Simplex Multiplexer

Recast to VCR Play

Diagram illustrating a video recording system using a Simplex Multiplexer and a VCR.

27

Point-to-Point Data Concentration

RS-232C Data Source

SDC Data Control

Modem or DSU

Simpler or Duplex Link

Modem or DSU

SDC Data Control

RS-232C Data USER

VCL-MX E1, 2Mbps Voice & Data Multiplexer

Connecting at the Central Office / Switch - E & M Interfaces

Central Office / Switch

VCL-MX E1, 2 Mbps Voice & Data Multiplexer

1 to 30 E & M Channels

Central Office / Switch

VCL-MX E1, 2 Mbps Voice & Data Multiplexer

1 to 30 E & M Channels

Up to 32 PCs or terminals on one line

Multiple PCs or terminals

Modems

Multiple, expensive data lines

Modems

Host or Server

Modem or DSU

Mux

Mux

Link speeds to 64 Kbps

DCB Muxes correct all link errors

Network management built into every DCB multiplexer:

Example

28

28

K-Channel multiplexing/demultiplexing

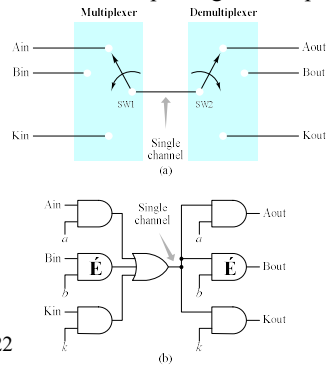
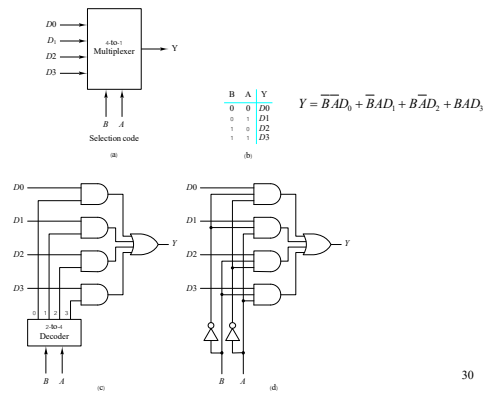


Figure 4.22

29

Four-to-one multiplexer design



30

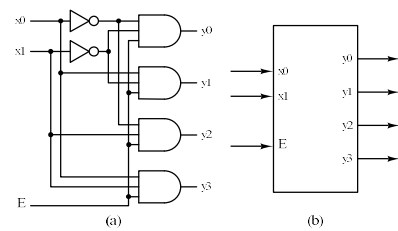
N4	N3	N2	N1	A2	A1	A0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	1	0
1	0	1	1	1	1	1
1	1	0	0	0	0	0
1	1	0	1	0	0	1
1	1	1	0	0	1	0
1	1	1	1	0	1	1

(b)

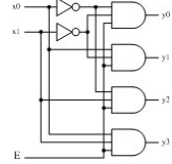
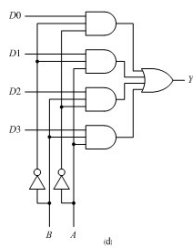
31

one-to-four demultiplexer design

- (a) Functional diagram
(b) Truth table

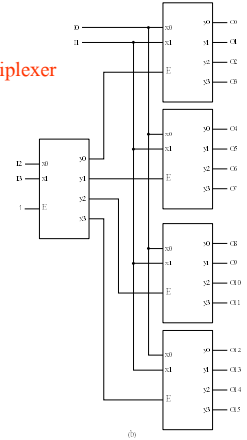


32

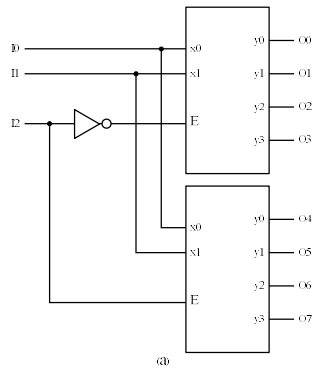


33

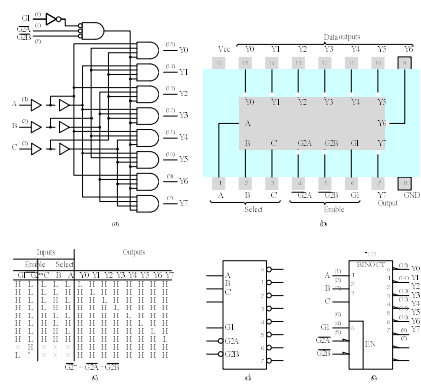
1-to-16 Demultiplexer



3-to-8 Decoder



35



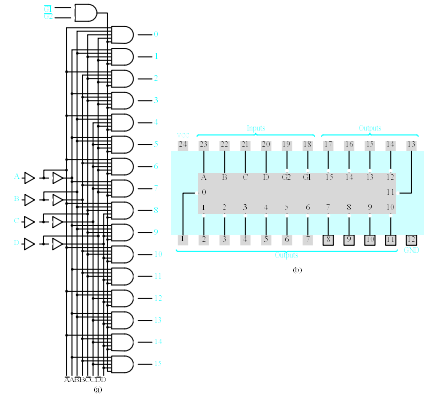
36

Inputs					Outputs							
Enable		Select										
G1	G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	L	H	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L
x	H	x	x	x	H	H	H	H	H	H	H	H
L	'	x	x	x	H	H	H	H	H	H	H	H

$$\overline{G2}^* = \overline{G2}A + \overline{G2}B$$

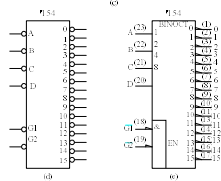
(C)

37

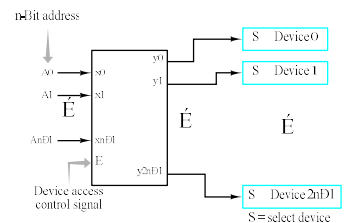


38

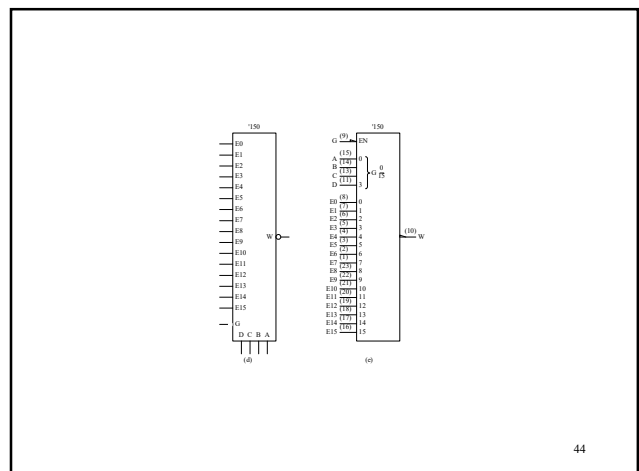
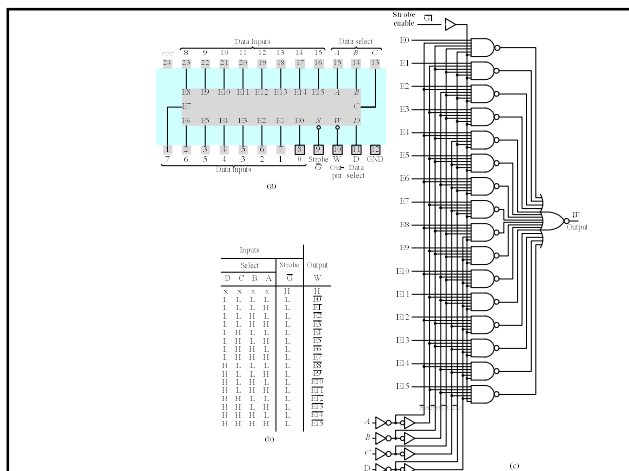
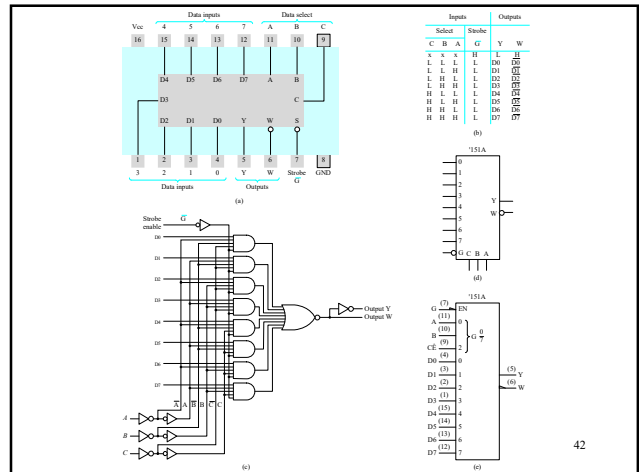
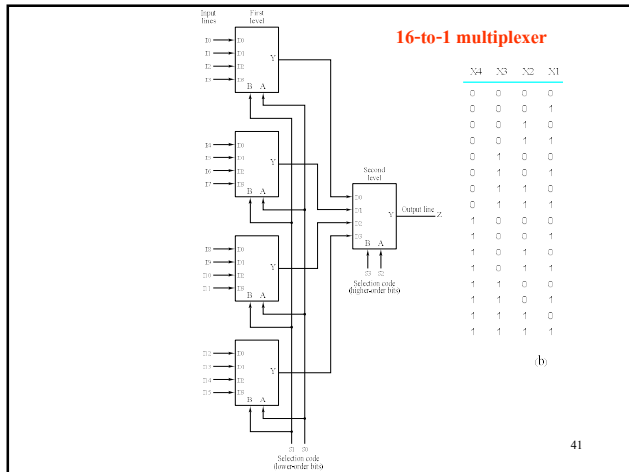
Inputs					Outputs													
A	B	C	D	E	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Y11	Y12	Y13
L	L	L	L	L	1	0	0	0	0	0	0	0	0	0	0	0	0	0
L	L	L	L	H	0	1	0	0	0	0	0	0	0	0	0	0	0	0
L	L	L	H	L	0	0	1	0	0	0	0	0	0	0	0	0	0	0
L	L	L	H	H	0	0	0	1	0	0	0	0	0	0	0	0	0	0
L	L	H	L	L	0	0	0	0	1	0	0	0	0	0	0	0	0	0
L	L	H	L	H	0	0	0	0	0	1	0	0	0	0	0	0	0	0
L	L	H	H	L	0	0	0	0	0	0	1	0	0	0	0	0	0	0
L	L	H	H	H	0	0	0	0	0	0	0	1	0	0	0	0	0	0
L	H	L	L	L	0	0	0	0	0	0	0	0	1	0	0	0	0	0
L	H	L	L	H	0	0	0	0	0	0	0	0	0	1	0	0	0	0
L	H	L	H	L	0	0	0	0	0	0	0	0	0	0	1	0	0	0
L	H	L	H	H	0	0	0	0	0	0	0	0	0	0	0	1	0	0
L	H	H	L	L	0	0	0	0	0	0	0	0	0	0	0	0	1	0
L	H	H	L	H	0	0	0	0	0	0	0	0	0	0	0	0	0	1
L	H	H	H	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	H	H	H	H	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	L	L	L	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	L	L	L	H	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	L	L	H	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	L	L	H	H	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	L	H	L	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	L	H	L	H	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	L	H	H	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	L	H	H	H	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	H	L	L	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	H	L	L	H	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	H	L	H	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	H	L	H	H	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	H	H	L	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	H	H	L	H	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	H	H	H	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	H	H	H	H	0	0	0	0	0	0	0	0	0	0	0	0	0	0



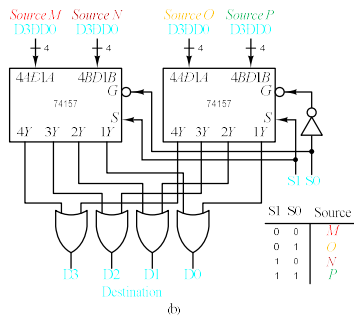
39



40







49

Use a 74151A multiplexer to Realize
 $f(x_1, x_2, x_3) = \sum m(0, 2, 3, 5)$

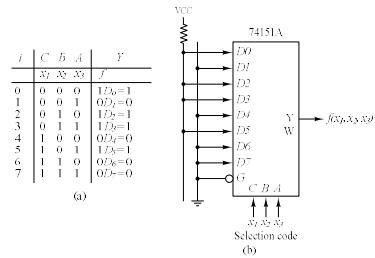
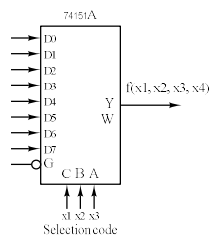


Figure 4.30

50

Use a 74151A multiplexer to Realize
 $f(x_1, x_2, x_3, x_4) = \sum m(0, 1, 2, 3, 4, 9, 13, 14, 15)$

i	C	B	A		
i	X1	X2	X3	X4	f
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

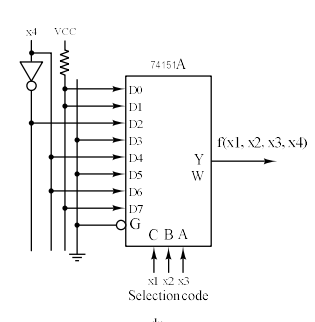


(a)

51

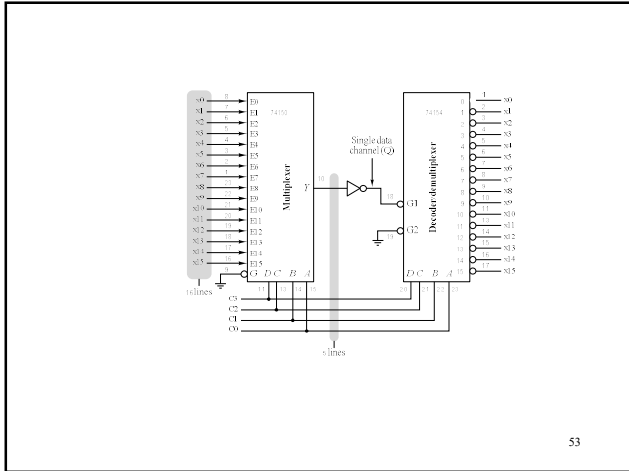
Use a 74151A multiplexer to Realize
 $f(x_1, x_2, x_3, x_4) = \sum m(0, 1, 2, 3, 4, 9, 13, 14, 15)$

i	C	B	A			
i	X1	X2	X3	X4	f	f
0	0	0	0	0	1	D0 = 1
1	0	0	0	1	1	D1 = 1
2	0	0	1	0	1	D2 = 1
3	0	0	1	1	1	D3 = 1
4	0	1	0	0	1	D4 = 1
5	0	1	0	1	0	D5 = 0
6	0	1	1	0	0	D6 = 0
7	0	1	1	1	0	D7 = 0
8	1	0	0	0	0	D8 = 0
9	1	0	0	1	1	D9 = 1
10	1	0	1	0	0	D10 = 0
11	1	0	1	1	1	D11 = 1
12	1	1	0	0	0	D12 = 0
13	1	1	0	1	1	D13 = 1
14	1	1	1	0	1	D14 = 1
15	1	1	1	1	1	D15 = 1

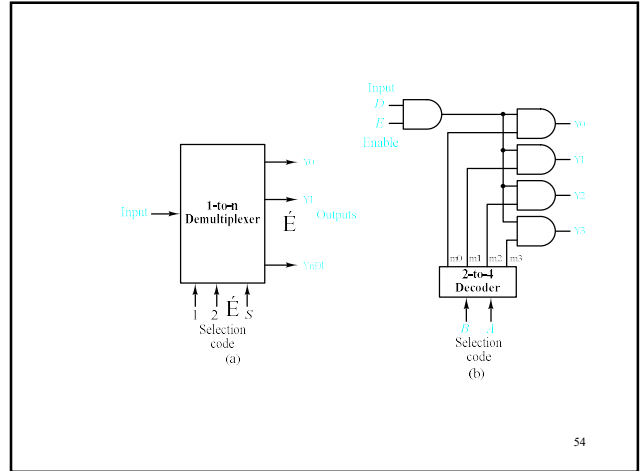


(a)

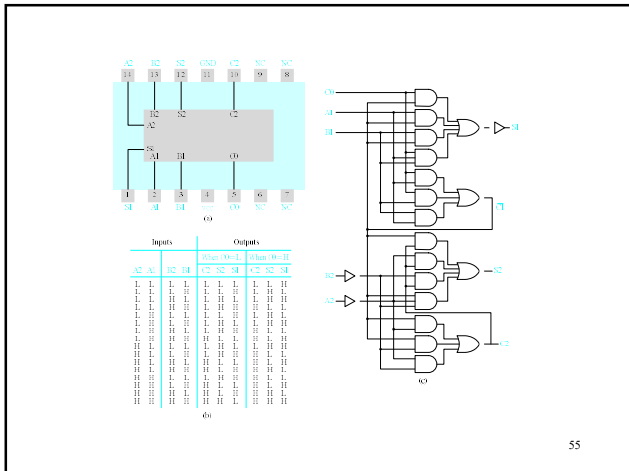
52



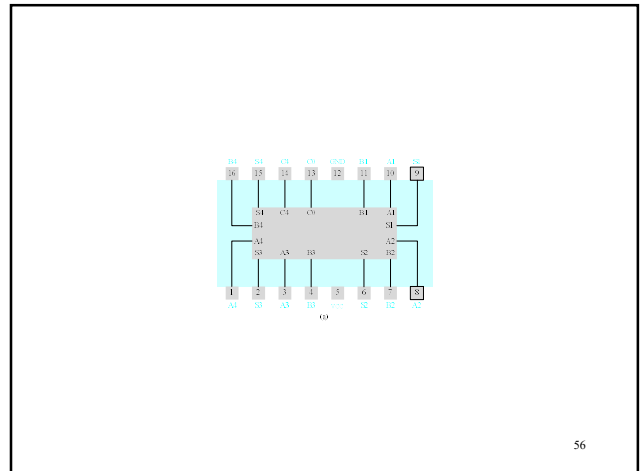
53



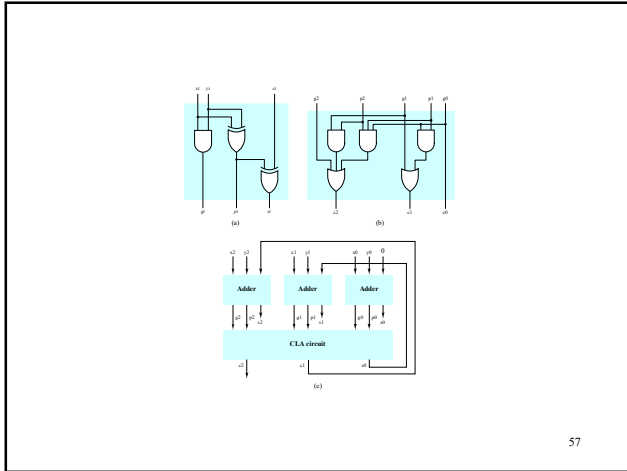
54



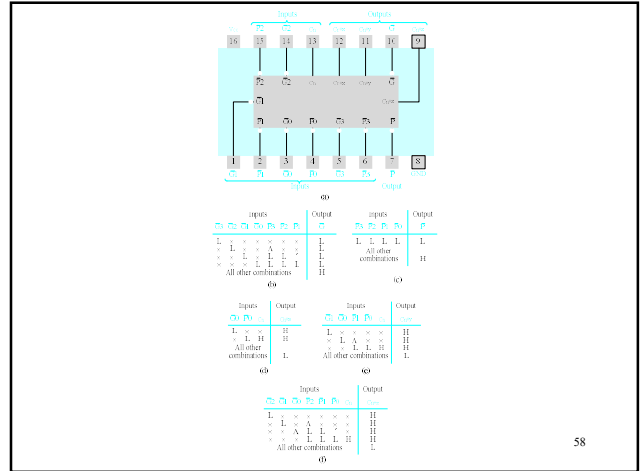
55



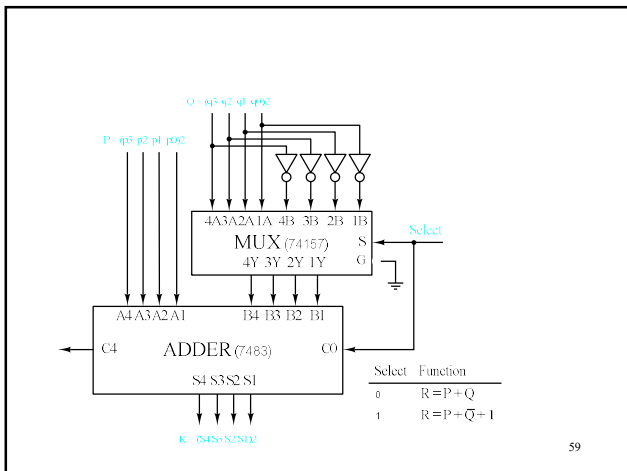
56



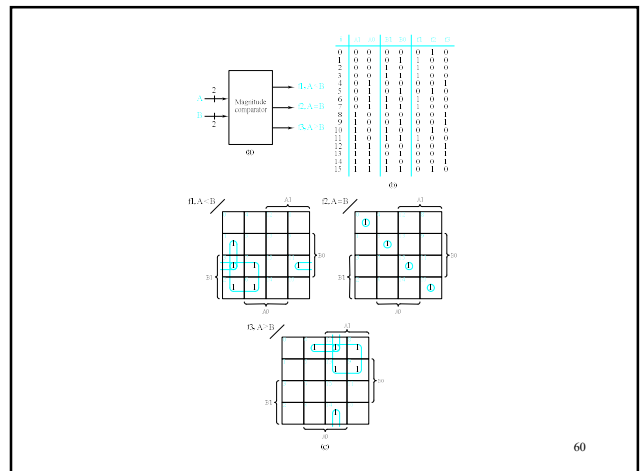
57



58



59

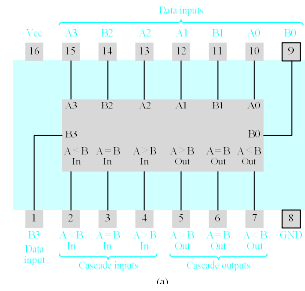


60

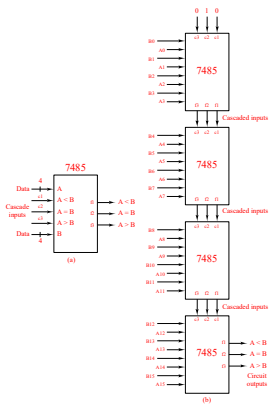
Comparing inputs				Cascading inputs			Outputs		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	×	×	×	×	×	×	H	L	L
A3 < B3	×	×	×	×	×	×	L	H	L
A3 = B3	A2 > B2	×	×	×	×	×	H	L	L
A3 = B3	A2 < B2	×	×	×	×	×	L	H	L
A3 = B3	A2 = B2	A1 > B1	×	×	×	×	H	L	L
A3 = B3	A2 = B2	A1 < B1	×	×	×	×	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	×	×	×	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	×	×	×	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	×	×	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

(b)

61



62



63

Half Adders

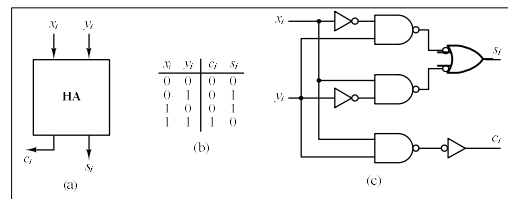
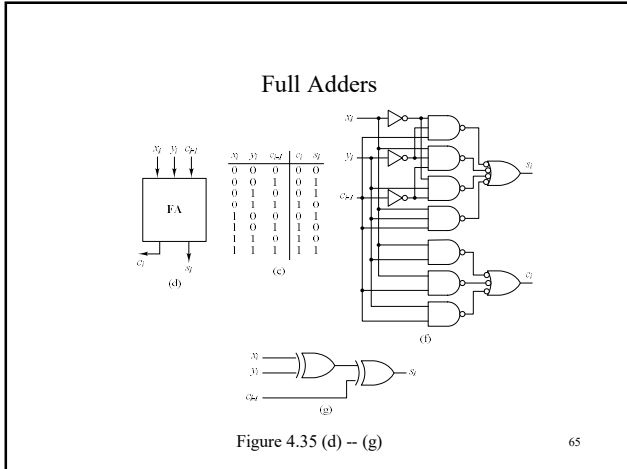
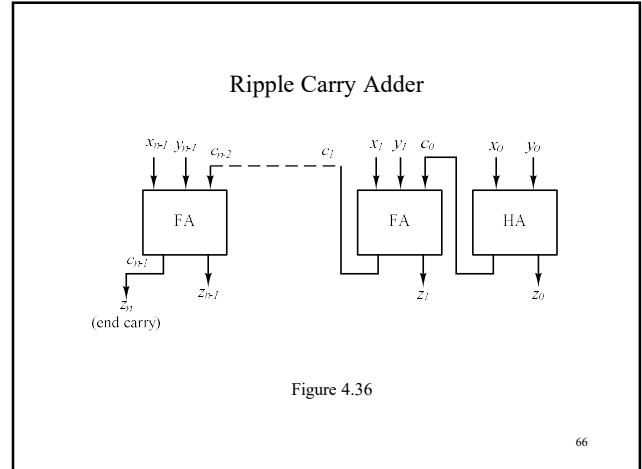


Figure 4.35 (a) -- (c)

64



65



66

Addition Time for a Basic Ripple-Carry Adder

Let t_{gate} = the propagation delay through a typical logic gate

Half adder propagation delays

$$t_{\text{add}} = 3 t_{\text{gate}}$$

$$t_{\text{carry}} = 2 t_{\text{gate}}$$

Full adder propagation delays

$$t_{\text{add}} = 3 t_{\text{gate}}$$

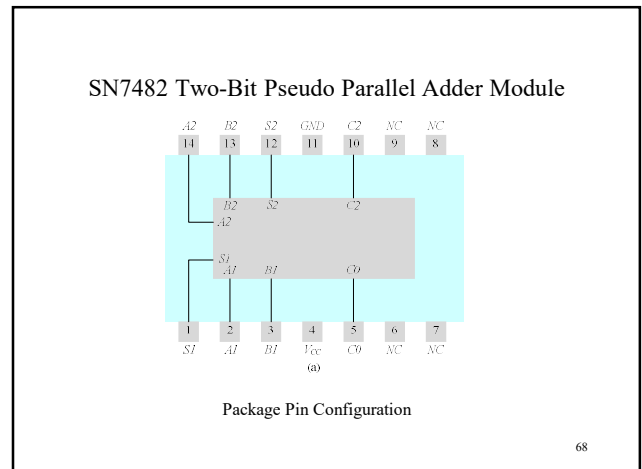
$$t_{\text{carry}} = 2 t_{\text{gate}}$$

Ripple-Carry Adder (n -bits)

$$t_{\text{add}} = (n - 1) 2 t_{\text{gate}} + 3 t_{\text{gate}}$$

$$= (2n + 1) t_{\text{gate}}$$

67



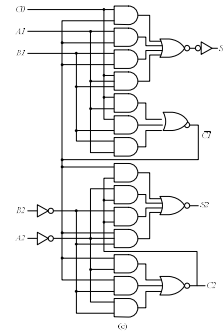
SN7482 Pseudo Parallel Adder -- Truth Table

Inputs				Outputs					
				When $C0=L$			When $C0=H$		
$A2$	$A1$	$B2$	$B1$	$C2$	$S2$	$S1$	$C2$	$S2$	$S1$
L	L	L	L	L	L	L	L	L	H
L	L	L	H	L	L	H	L	H	L
L	L	H	L	L	H	L	L	H	H
L	L	H	H	L	H	H	H	L	L
L	H	L	L	L	L	H	L	H	L
L	H	L	H	L	H	L	L	H	H
L	H	H	L	L	H	H	H	L	L
L	H	H	H	L	H	H	H	L	H
H	L	L	L	L	L	L	L	H	H
H	L	L	H	L	H	H	H	L	L
H	L	H	L	L	L	L	H	L	L
H	L	H	H	L	H	L	H	L	H
H	H	L	L	L	H	H	H	L	L
H	H	L	H	L	L	L	H	L	H
H	H	H	L	L	H	L	H	L	H
H	H	H	H	L	H	H	H	L	L

(b)

69

SN7482 Pseudo Parallel Adder -- Logic Diagram



70

SN7482 Two-Bit Adder -- Logic Equations

$$C1 = C0A1 + C0B1 + A1B1 \quad (4.20)$$

$$\begin{aligned} \Sigma1 &= C0C1' + A1C1' + B1C1' + A1B1C0 \\ &= C1'(C0 + A1 + B1) + A1B1C0 \\ &= (C0' + A1')(C0' + B1')(A1' + B1')(C0 + A1 + B1) + A1B1C0 \\ &= (C0' + A1'B1')(A1' + B1')(C0 + A1 + B1) + A1B1C0 \quad (4.21) \\ &= [C0'(A1 + B1) + C0A1'B1'](A1' + B1') + A1B1C0 \\ &= C0'A1B1' + C0'A1'B1 + C0A1'B1' + A1B1C0 \\ &= C0 \oplus A1 \oplus B1 \end{aligned}$$

Similarly

$$C2 = C1A2 + C1B2 + A2B2 \quad (4.22)$$

$$\Sigma2 = C1 \oplus A2 \oplus B2$$

71

Add Time for SN7482 Adder Circuits

SN7482 propagation delays

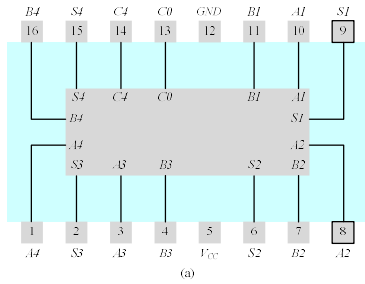
$$\begin{aligned} t_{\Sigma1} &= 5 t_{\text{gate}} \\ t_{C1} &= 2 t_{\text{gate}} \\ t_{\Sigma2} &= 6 t_{\text{gate}} \\ t_{C2} &= 4 t_{\text{gate}} \end{aligned}$$

SN7482-based ripple-carry adder (n -bits)

$$t_{\text{add}} = (2n + 2) t_{\text{gate}}$$

72

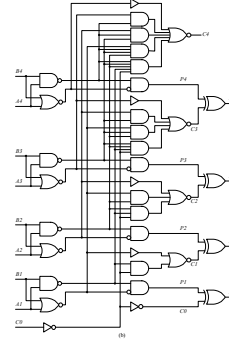
SN7483 Four-Bit Adder Module



Package Pin Configuration

73

SN7483 Four-Bit Adder Module -- Logic Diagram



74

SN7483 Four-Bit Adder -- Logic Equations

$$\begin{aligned} P_i &= (B_i A_i)'(A_i + B_i) \\ &= (A_i' + B_i')(A_i + B_i) \\ &= A_i \oplus B_i \end{aligned} \quad (4.24)$$

$$\begin{aligned} \Sigma_i &= P_i \oplus C_{i-1} \\ &= A_i \oplus B_i \oplus C_{i-1} \end{aligned} \quad (4.25)$$

$$\begin{aligned} C_1 &= [C_0'(A_1 B_1)' + (A_1 + B_1)']' \\ &= [C_0'(A_1 B_1)']'(A_1 + B_1) \\ &= (C_0 + (A_1 B_1))(A_1 + B_1) \\ &= C_0 A_1 + C_0 B_1 + A_1 B_1 \end{aligned} \quad (4.26)$$

Similarly

$$C_i = C_{i-1} A_i + C_{i-1} B_i + A_i B_i$$

75

Add Times for SN7483 Adder Circuits

SN7483 propagation delays

$$\begin{aligned} t_{\Sigma 1} &= 3 t_{\text{gate}} \\ t_{\Sigma 2} &= t_{\Sigma 3} = t_{\Sigma 4} = 4 t_{\text{gate}} \\ t_{C1} &= t_{C2} = t_{C3} = t_{C4} = 3 t_{\text{gate}} \end{aligned}$$

SN7483-based Ripple-Carry Adder (n -bits)

$$t_{\text{add}} = (3m + 1) t_{\text{gate}}$$

where $m = \lceil n/4 \rceil$.

76

Fully Parallel Three-Bit Adder

$$\begin{aligned} c_0 &= x_0 y_0 \\ s_0 &= x_0 \oplus y_0 \end{aligned} \quad (4.30)$$

$$\begin{aligned} c_1 &= x_1 y_1 c_0' + x_1 y_1 c_0 + x_1 y_1' c_0' + x_1' y_1 c_0 \\ &= x_1 y_1 + (x_1 \oplus y_1) c_0 \\ &= x_1 y_1 + (x_1 \oplus y_1)(x_0 y_0) \\ s_1 &= x_1 \oplus y_1 \oplus c_0 \\ &= x_1 \oplus y_1 \oplus x_0 y_0 \end{aligned} \quad (4.31)$$

$$\begin{aligned} c_2 &= x_2 y_2 + (x_2 \oplus y_2) c_1 \\ &= x_2 y_2 + (x_2 \oplus y_2)[x_1 y_1 + (x_1 \oplus y_1)(x_0 y_0)] \\ &= x_2 y_2 + (x_2 \oplus y_2)(x_1 y_1) + (x_2 \oplus y_2)(x_1 \oplus y_1)(x_0 y_0) \\ s_2 &= x_2 \oplus y_2 \oplus c_1 \\ &= x_2 \oplus y_2 \oplus [x_1 y_1 + (x_1 \oplus y_1)(x_0 y_0)] \end{aligned} \quad (4.32)$$

77

Add Time for a Fully Parallel Adder

Assuming a three-level realization

$$t_{\text{add}} = 3 t_{\text{gate}}$$

However, the fan in requirements become impractical as n increases.

78

Carry Look-Ahead Adders -- Basic Idea

Recall that

$$\begin{aligned} c_i &= x_i y_i + x_i c_{i-1} + y_i c_{i-1} \\ &= x_i y_i + x_i y_i' c_{i-1} + x_i y_i' c_{i-1} + x_i y_i c_{i-1} + x_i' y_i c_{i-1} \\ &= x_i y_i + x_i y_i' c_{i-1} + x_i' y_i c_{i-1} \\ &= x_i y_i + (x_i y_i' + x_i' y_i) c_{i-1} \\ &= x_i y_i + (x_i \oplus y_i) c_{i-1} \end{aligned}$$

Let $g_i = x_i y_i$ [carry generate] (4.33)

$p_i = x_i \oplus y_i$ [carry propagate] (4.34)

Then $c_i = g_i + p_i c_{i-1}$ (4.38)

79

Carry Look-Ahead Adders -- Three-Bit Example

$$\begin{aligned} c_0 &= g_0 \\ s_0 &= p_0 \end{aligned} \quad (4.35)$$

$$\begin{aligned} c_1 &= g_1 + p_1 c_0 \\ &= g_1 + p_1 g_0 \\ s_1 &= p_1 \oplus c_0 \end{aligned} \quad (4.36)$$

$$\begin{aligned} c_2 &= g_2 + p_2 c_1 \\ &= g_2 + p_2 (g_1 + p_1 g_0) \\ &= g_2 + p_2 g_1 + p_2 p_1 g_0 \\ s_2 &= p_2 \oplus c_1 \end{aligned} \quad (4.37)$$

80

Carry Look-Ahead Adder Design

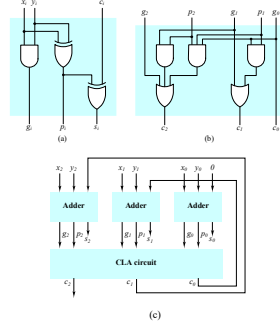


Figure 4.39

81

Add Times for Carry Look-Ahead Adders

Adder modules

$$t_g = t_p = t_s = t_{\text{gate}}$$

CLA module

$$t_c = 2 t_{\text{gate}}$$

Overall

$$\begin{aligned} t_{\text{add}} &= t_{\text{gate}} + 2 t_{\text{gate}} + t_{\text{gate}} \\ &= 4 t_{\text{gate}} \end{aligned}$$

82

Binary Subtraction Circuits

Recall that

$$\begin{aligned} (R)_2 &= (P)_2 - (Q)_2 \\ &= (P)_2 + (-Q)_2 \\ &= (P)_2 + [(Q)_2] \\ &= (P)_2 + (Q)_2' + 1 \end{aligned}$$

For an SN7483 adder

$$(\Sigma)_2 = (A)_2 + (B)_2 + (C0)_2 \quad (4.39)$$

where $\Sigma = \Sigma 4 \Sigma 3 \Sigma 2 \Sigma 1$, $A = A4 A3 A2 A1$, and $B = B4 B3 B2 B1$

If $C0 = 0$, $A = P$, and $B = Q$, then $(\Sigma)_2 = (P)_2 + (Q)_2$.

If $C0 = 1$, $A = P$, and $B = Q'$, then $(\Sigma)_2 = (P)_2 - (Q)_2$.

83

Two's Complement Adder/Subtractor

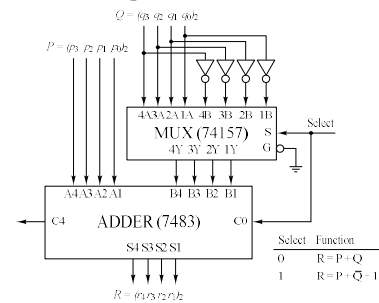


Figure 4.41

84

Arithmetic Overflow Detection

a_{n-1}	b_{n-1}	c_{n-2}	c_{n-1}	s_{n-1}	V
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0

85

Overflow Detection Circuits

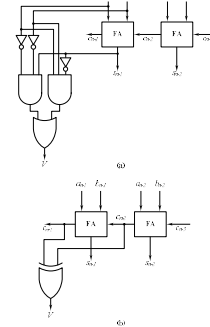
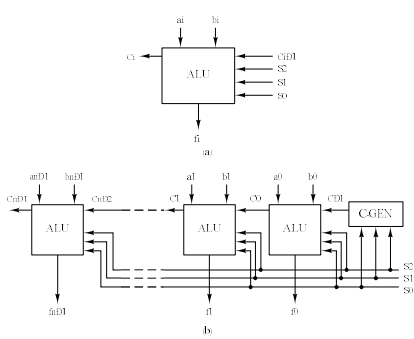
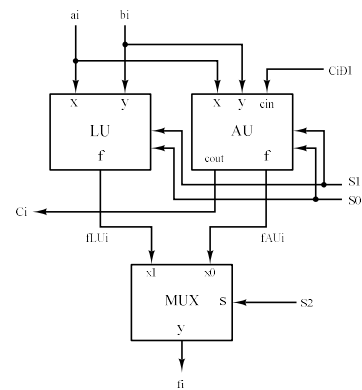


Figure 4.42

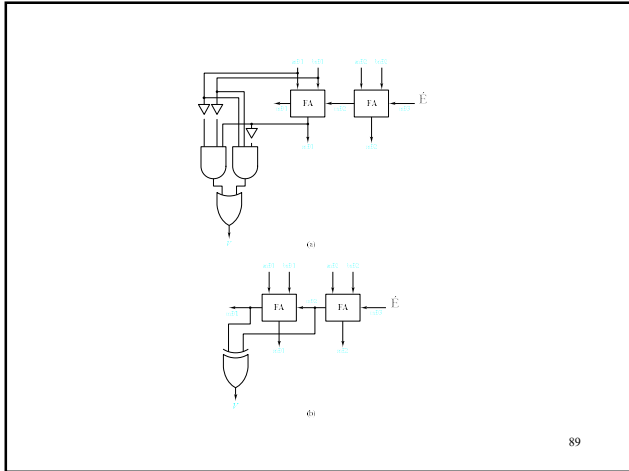
86



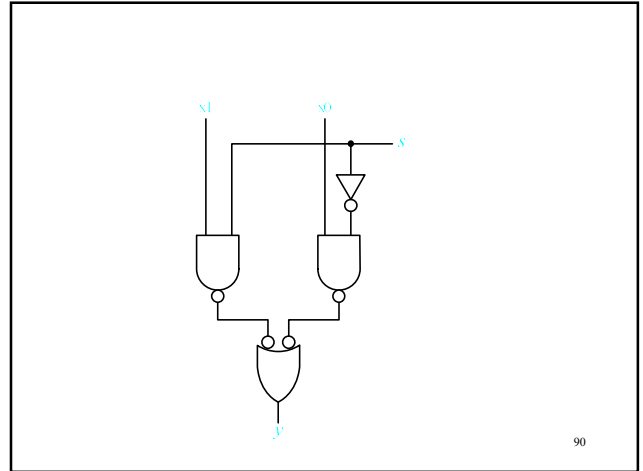
87



88



89

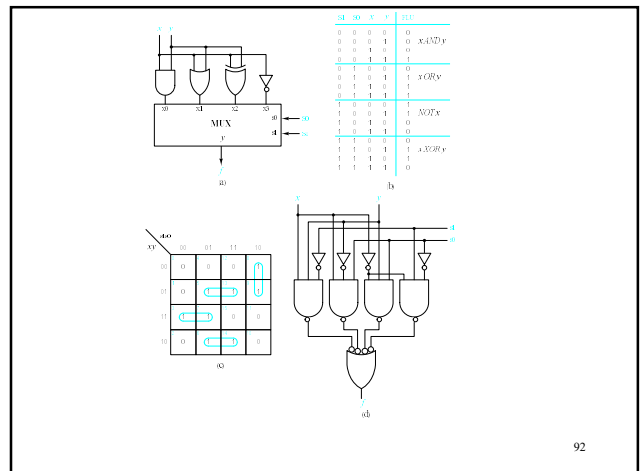


90

s_1	s_0	x	y	FLU
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

(b)

91



92

SI	SO	x	y	FLU
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

93

SI	SO	bi	yi	
0	0	0	0	Add
0	0	1	1	
0	1	0	1	Subtract
0	1	1	0	
1	0	0	0	Increment
1	0	1	0	
1	1	0	1	Decrement
1	1	1	1	

(a)

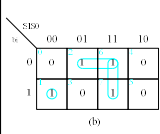
94

SI	SO	bi	yi	
0	0	0	0	Add
0	0	1	1	
0	1	0	1	Subtract
0	1	1	0	
1	0	0	0	Increment
1	0	1	0	
1	1	0	1	Decrement
1	1	1	1	

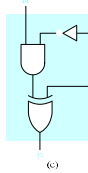
(a)

SI	SO	CTD	
C	0	0	Add
C	1	1	Subtract
1	0	1	Increment
1	1	0	Decrement

(a)



(b)

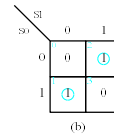


(c)

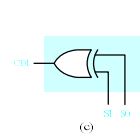
95

SI	SO	CTD	
0	0	0	Add
0	1	1	Subtract
1	0	1	Increment
1	1	0	Decrement

(a)

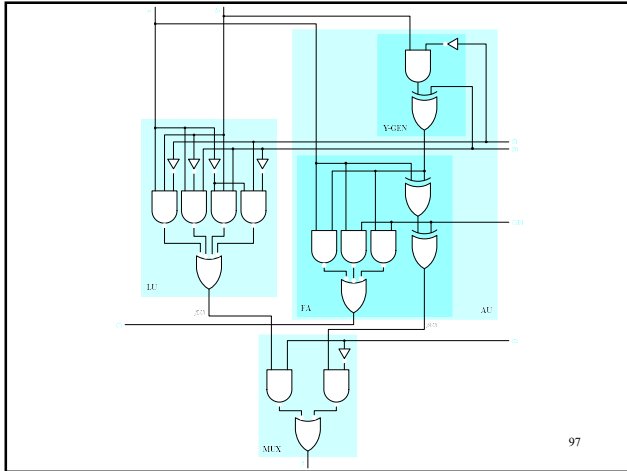


(b)

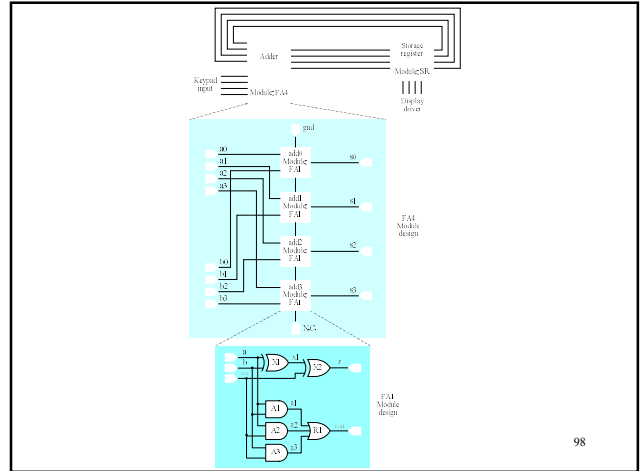


(c)

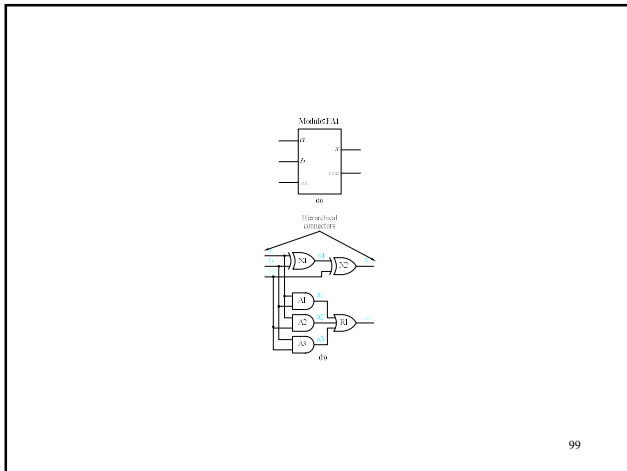
96



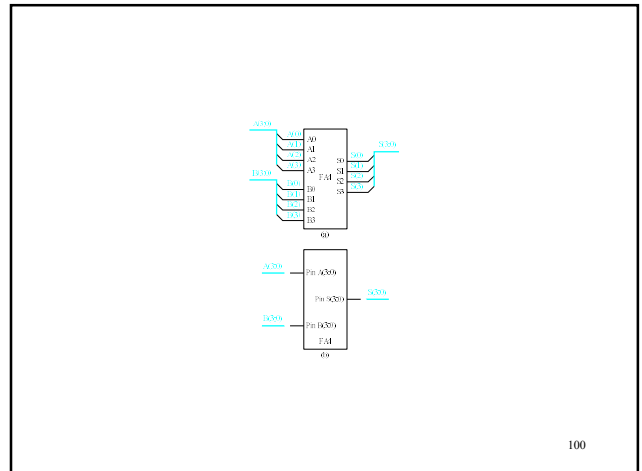
97



98



99



100

