

# Chapter 6

## Combinational Case Studies

### SKEE2263 Digital Systems

Mun'im Zabidi {munim@utm.my}  
Ismahani Ismail {ismahani@fke.utm.my}  
Izam Kamisian {e-izam@utm.my}

Faculty of Electrical Engineering, Universiti Teknologi Malaysia

March 30, 2018

# Table of Contents

- 1 Design Procedure
- 2 BCD Adder
- 3 Binary to BCD Converter
- 4 Barrel Shifter
- 5 Comparator
- 6 Incrementer

# Design Procedure

<b>Step</b>	<b>Description</b>
<i>Specification</i>	Write a specification for the circuit.
<i>Formulation</i>	Derive the truth table or initial Boolean equations that define the required relationships between inputs and outputs.
<i>Optimization</i>	Apply two-level and multi-level optimization.
<i>Design entry</i>	Draw the schematic for the resulting circuit or key in the hardware design language code.
<i>Technology Mapping</i>	Transform the logic diagram to a new diagram using the available implementation technology.
<i>Verification</i>	Verify the correctness of the final design.

# The Problem with BCD

$$\begin{array}{r} 0110 \\ + 0011 \\ \hline \end{array} = \begin{array}{r} 6 \\ + 3 \\ \hline 9 \end{array}$$

1001 = 9

Correct:  
Result is in BCD

$$\begin{array}{r} 0101 \\ + 0111 \\ \hline \end{array} = \begin{array}{r} 5 \\ + 7 \\ \hline 12 \end{array}$$

1100 = 12

Wrong:  
Result is not in BCD

## Correction Factor (when sum > 9)

	0101	5
	+ 0111	+ 7
Non BCD	1100	12
Adjustment	+ 0110	+ 6
Correct BCD	1 0010	1 2

	1000	8
	+ 0111	+ 7
Non BCD	1111	15
Adjustment	+ 0110	+ 6
Correct BCD	1 0101	1 5

	1001	9
	+ 1001	+ 9
Non BCD	1 0010	18
Adjustment	+ 0110	+ 6
Correct BCD	1 1000	1 8

# All Possible Adder Outputs

Decimal	Binary Sum					BCD Sum				
	$C$	$s_3$	$s_2$	$s_1$	$s_0$	$K$	$z_3$	$z_2$	$z_1$	$z_0$
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

## Correction Factor

When the sum exceeds 9:

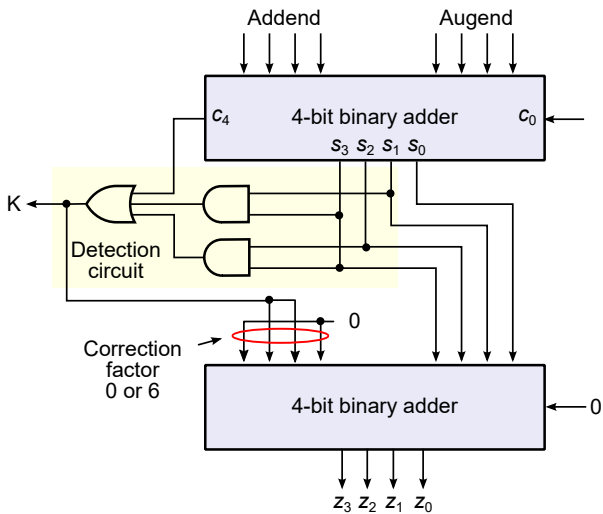
- Carry bit  $C$  is 1, or
- The binary sum is between 1010 and 1111

		$s_1s_0$			
		00	01	11	10
$s_3s_2$	00				
	01				
	11	1	1	1	1
	10			1	1

The correction step (adding 0110) is performed if  $C$  or  $f$  is true:

$$K = C + s_3s_2 + s_3s_1$$

# BCD Adder Circuit





## Double-dabble algorithm

For an  $n$ -bit binary number, the steps are:

- 1 Find the number of bits which holds the biggest BCD output.
- 2 Divide the output into BCD columns of 4 bits each
- 3 Shift the binary number one bit
- 4 If the binary number in any of the BCD columns is 5 or greater, add 3 to the BCD columns.
- 5 Repeat from Step 3 until all  $n$  bits have been shifted.

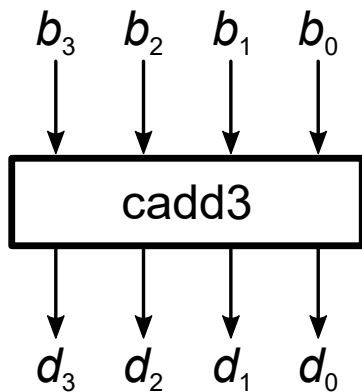
# 3F → 63

Operation	Tens	Ones	Binary
HEX			3F
Start			111111
Shift 1		1	11111
Shift 2		11	1111
Shift 3		111	111
Add 3		1010	111
Shift 4	1	0101	11
Add 3	1	1000	11
Shift 5	11	0001	1
Shift 6	110	0011	
BCD	6	3	

FF → 255

Operation	Hundreds	Tens	Ones	Binary	
Start				1111	1111
Shift 1			1	1111	111
Shift 2			11	1111	11
Shift 3			111	1111	1
Add 3			1010	1111	1
Shift 4		1	0101	1111	
Add 3		1	1000	1111	
Shift 5		11	0001	111	
Shift 6		110	0011	11	
Add 3		1001	0011	11	
Shift 7	1	0010	0111	1	
Add 3	1	0010	1010	1	
Shift 8	10	0101	0101		

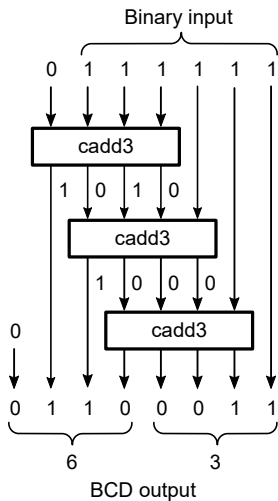
## *add3* Module



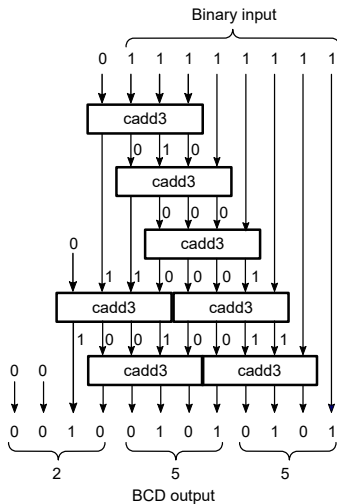
# add3 Module

Binary Input				Decimal Output				Function
$b_3$	$b_2$	$b_1$	$b_0$	$d_3$	$d_2$	$d_1$	$d_0$	
0	0	0	0	0	0	0	0	Add 0
0	0	0	1	0	0	0	1	
0	0	1	0	0	0	1	0	
0	0	1	1	0	0	1	1	
0	1	0	0	0	1	0	0	
0	1	0	1	1	0	0	0	Add 3
0	1	1	0	1	0	0	1	
0	1	1	1	1	0	1	0	
1	0	0	0	1	0	1	1	
1	0	0	1	1	1	0	0	
1	0	1	0	x	x	x	x	Don't care
1	0	1	1	x	x	x	x	
1	1	0	0	x	x	x	x	
1	1	0	1	x	x	x	x	
1	1	1	0	x	x	x	x	
1	1	1	1	x	x	x	x	

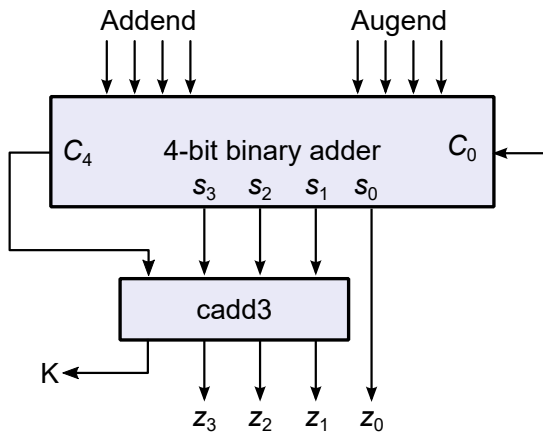
## 6 bit binary $\rightarrow$ 2 BCD digits



# 8 bit binary $\rightarrow$ 3 BCD digits

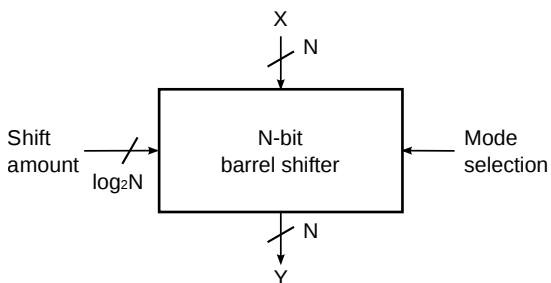


# Improved BCD Adder

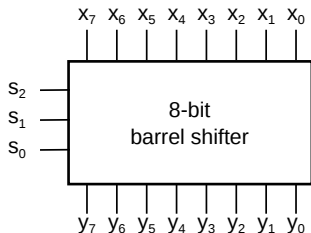




# Generic Barrel Shifter

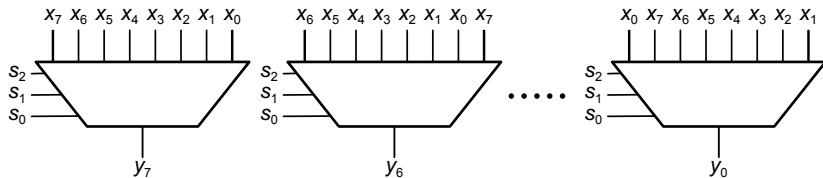


# Barrel Shifter Outputs

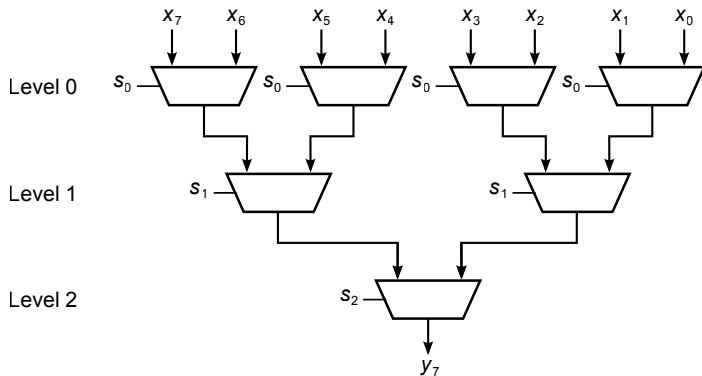


Shift amount	$y_7$	$y_6$	$y_5$	$y_4$	$y_3$	$y_2$	$y_1$	$y_0$
0	$x_7$	$x_6$	$x_5$	$x_4$	$x_3$	$x_2$	$x_1$	$x_0$
1	$x_6$	$x_5$	$x_4$	$x_3$	$x_2$	$x_1$	$x_0$	$x_7$
2	$x_5$	$x_4$	$x_3$	$x_2$	$x_1$	$x_0$	$x_7$	$x_6$
4	$x_3$	$x_2$	$x_1$	$x_0$	$x_7$	$x_6$	$x_5$	$x_4$
7	$x_0$	$x_7$	$x_6$	$x_5$	$x_4$	$x_3$	$x_2$	$x_1$

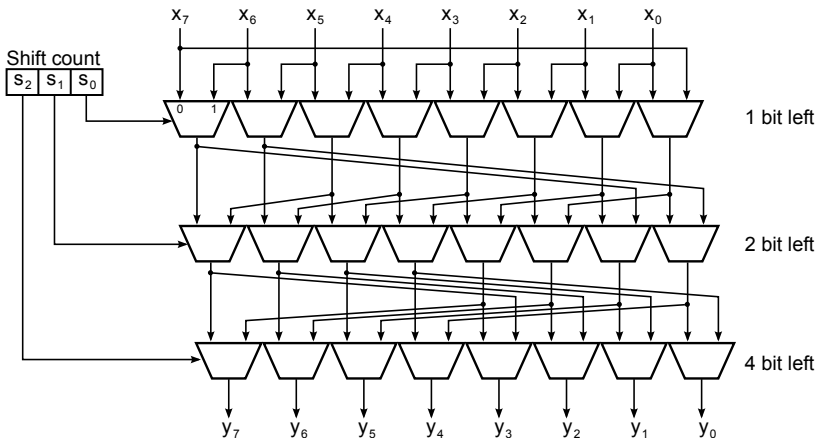
# Implementation by 8:1 MUX



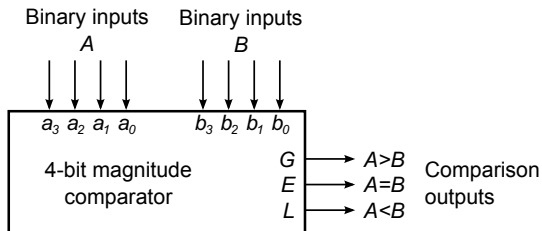
# Implementation by 2:1 MUX



# Eight-bit Barrel Shifter



# Magnitude Comparator Top-Level



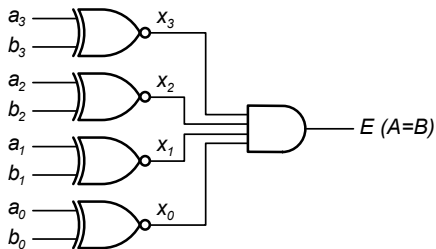
$$E = (A = B) = x_3 x_2 x_1 x_0$$

$$L = (A < B) = a'_3 b_3 + x_3 a'_2 b_2 + x_3 x_2 a'_1 b_1 + x_3 x_2 x_1 a'_0 b_0$$

$$G = (A > B) = a_3 b'_3 + x_3 a_2 b'_2 + x_3 x_2 a_1 b'_1 + x_3 x_2 x_1 a_0 b'_0$$

where  $x_i = a_i \odot b_i = a_i b_i + a'_i b'_i$  for  $i = 0, 1, 2, 3$ .

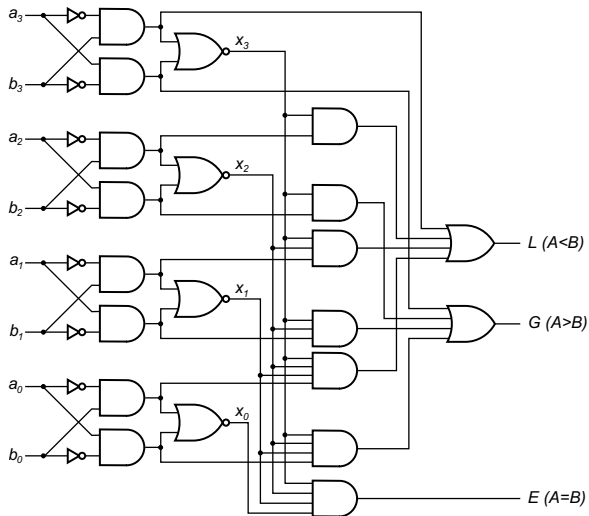
## 4-Bit Equality Comparator



$$E = (A = B) = x_3x_2x_1x_0$$

$$\text{where } x_i = a_i \odot b_i = a_i b_i + a'_i b'_i$$

# 4-Bit Comparator





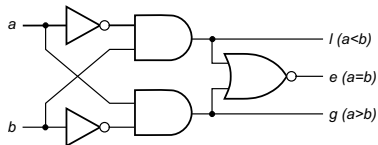
## Comparator Slice: End Module

$a$	$b$	$l$ $(a < b)$	$e$ $(a = b)$	$g$ $(a > b)$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

$$l = (a < b) = a'b$$

$$e = (a = b) = a'b' + ab = a \odot b$$

$$g = (a > b) = ab'$$

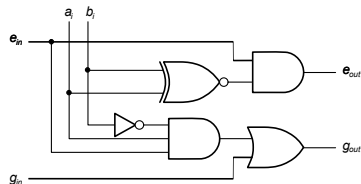


# Comparator Slice: Cascadable Module

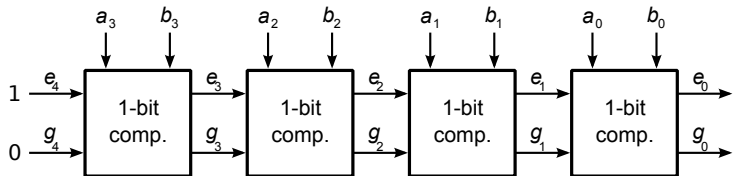
$e_{in}$	$g_{in}$	$a_i$	$b_i$	$e_{out}$	$g_{out}$
0	0	×	×	0	0
0	1	×	×	0	1
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	1	0

$$e_{out} = e_{in} \cdot (a_i \odot b_i)$$

$$g_{out} = g_{in} + e_{in} \cdot a_i \cdot b_i'$$



## 4-Bit Iterative Comparator



## 4-Bit Incrementer

