Lecture 15
EE 421 / CS 425
Digital System Design

Fall 2024

Shahid Masud

Topics

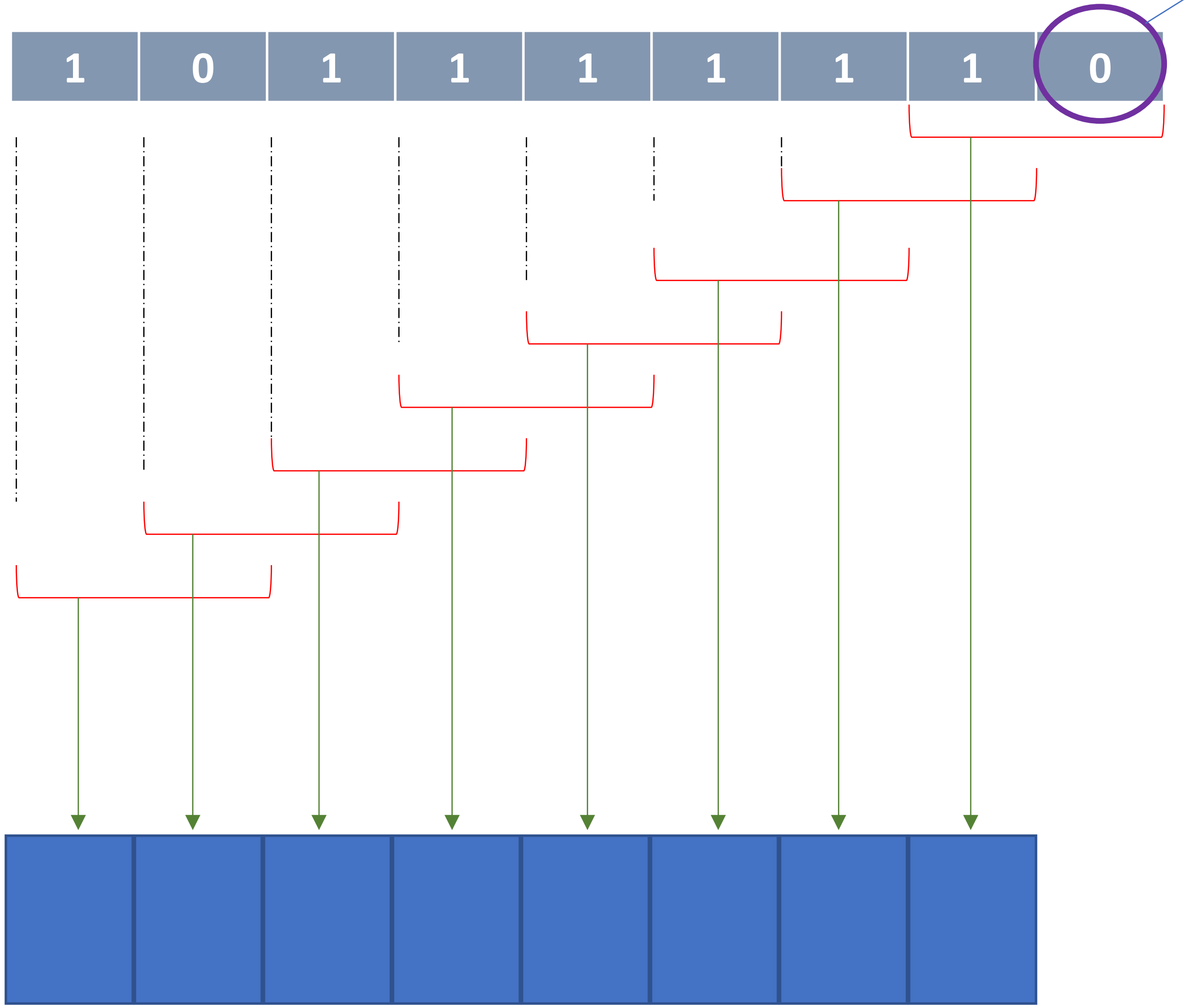
Midterm next Monday 28 October

- Booth Encoding and Booth Multiplication - Recap
- Modified Booth / Radix 4 Conversion
- Booth and Radix 4 Multiplication Process
- Booth and Radix 4 Multiplication Examples
- STG for Booth and Radix 4 Sequential Multipliers

Booth Recoding of -65_{10}

Append ‘0’ on right, if LSB=1

$-65_{10} =$



2’s Complement notation

+65 = (01000001)
2’s Complement
-65 = (10111111)

m_i	m_{i-1}	Booth Recoded Ci
0	0	0
0	1	1
1	0	$\underline{1}$
1	1	0

$-65_{10} =$

Or

Booth Recoded notation

Question?

Convert decimal number –78 to Booth Encoded format using 8 binary bits

1 0 1 1 0 0 1 0

No need for extra '0' after LSB in this case

+78 = 01001110
Take 2's Complement
-78 = 10110010

10 → -1
01 → 1
00 → 0
10 → -1
11 → 0
01 → 1
10 → -1

m_i	m_{i-1}	Booth Recoded Ci
0	0	0
0	1	1
1	0	1
1	1	0

Answer = -1 1 0 -1 0 1 -1

After Booth Encoding

Booth Multiplication – Example 1

m_i	m_{i-1}	Booth Recoded Ci	*Multiplication Use*
0	0	0	Only shift
0	1	1	Add, shift
1	0	$\underline{1}$	Sub, shift
1	1	0	Only shift

Show Booth Encoded multiplication of 6 x 5, using 4 bits for both numbers

Imagine Zero bit if LSB = 1

Check 2-bits at a time, Right to Left
Shift Left by 1 after every step

6	Multiplicand	Extra		0	0	0	0	0	1	1	0		
x5	Multiplier	bits		0	0	0	0	0	1	0	1	[0]	
				1	1	1	1	1	0	1	0		1[0] = Subtract = Add 2's Compl of Multiplicand to Acc
			0	0	0	0	0	1	1	0	X		01 = Add Multiplicand to Acc
		1	1	1	1	1	0	1	0	X	X		10 = Subtract = Add 2's Compl of Multiplicand to Acc
		0	0	0	0	1	1	0	X	X	X		01 = Add Multiplicand to Acc
								X	X	X	X		00 = No Op, Shift Left by 1
		0	1	0	0	0	1	1	1	1	0		00 = No Op, Shift Left by 1

Answer = (0001 1110) = +(16 + 14) = +30_{10}

Booth Multiplication – Example 2

Show Booth Encoded multiplication of 6 x -5, using 4 bits for both numbers

6	Multiplicand			0	0	0	0	0	1	1	0	
X -5	Multiplier			1	1	1	1	1	0	1	1	[0]
				1	1	1	1	1	0	1	0	
											X	
		0	0	0	0	0	1	1	0	X	X	
		1	1	1	1	0	1	0	X	X	X	
								X	X	X	X	
		0	0	1	1	1	0	0	0	1	0	

Imagine Zero bit if LSB = 1

Check 2-bits at a time, Right to Left
Shift Left by 1 after every step

1[0] = Subtract = Add 2’s Compl of Multiplicand to Acc

11 = No Op, Shift Left, Add 0 to Acc

01 = Add Multiplicand to Acc

10 = Subtract = Add 2’s Compl of Multiplicand to Acc

11 = No Op, Shift Left by 1, Add 0 to Acc

Answer = (1110 0010) = Take 2’s Comp = -(0001 1110) = -30_{10}

Booth Multiplication – Example 3

Show Booth Encoded multiplication of B3 x C3, using 8 bits for both numbers

B3 = 1011 0011 = 2’s Compl of = 0100 1101 = $(4D)_{16}$ = 77_{10}

C3 = 1100 0011 = 2’s Compl of = 0011 1101 = (3D) = 61_{10}

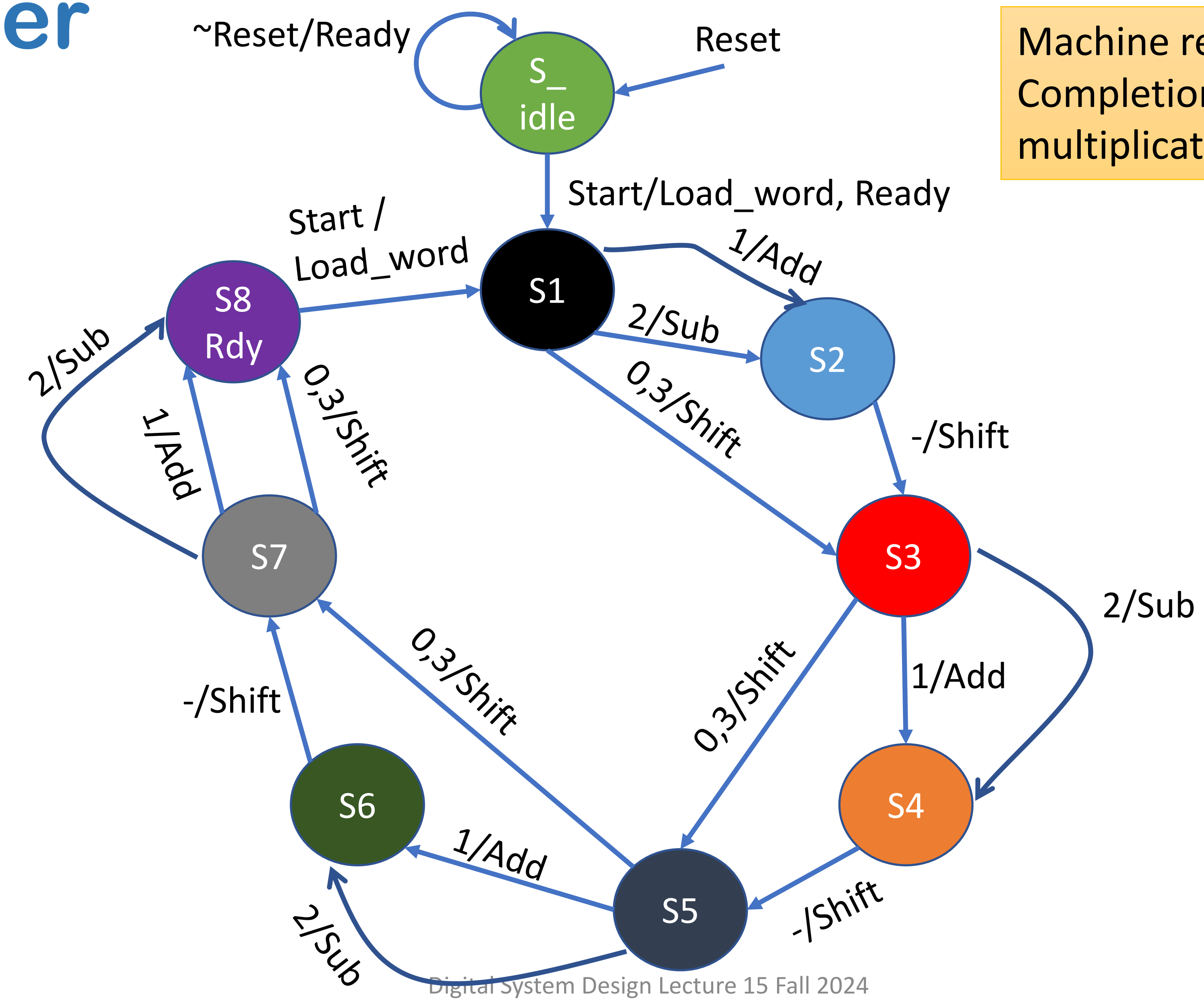
Imaginary Zero bit if LSB = 1

Check 2-bits at a time, Right to Left
Shift Left by 1 after every step

B3 Multiplicand										1	0	1	1	0	0	1	1		
X C3 Multiplier										1	1	0	0	0	0	1	1	[0]	
		0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1		1[0] = Subtract = Add 2’s Compl of Multiplicand to Acc
																	X		11 = No Op, Shift Left, Add 0 to Acc
		1	1	1	1	1	1	1	0	1	1	0	0	1	1	X	X		01 = Add Multiplicand to Acc
													X	X	X	X	X		00, 00, 00 = No Op, Shift Left by 1
		0	0	0	1	0	0	1	1	0	1	X	X	X	X	X	X		10 = Subtract = Add 2’s Compl of Multiplicand to Acc
	1	0	0	0	1	0	0	1	0	0	1	0	1	1	0	0	1		11 = No Op, Shift Left, Add 0 to Acc

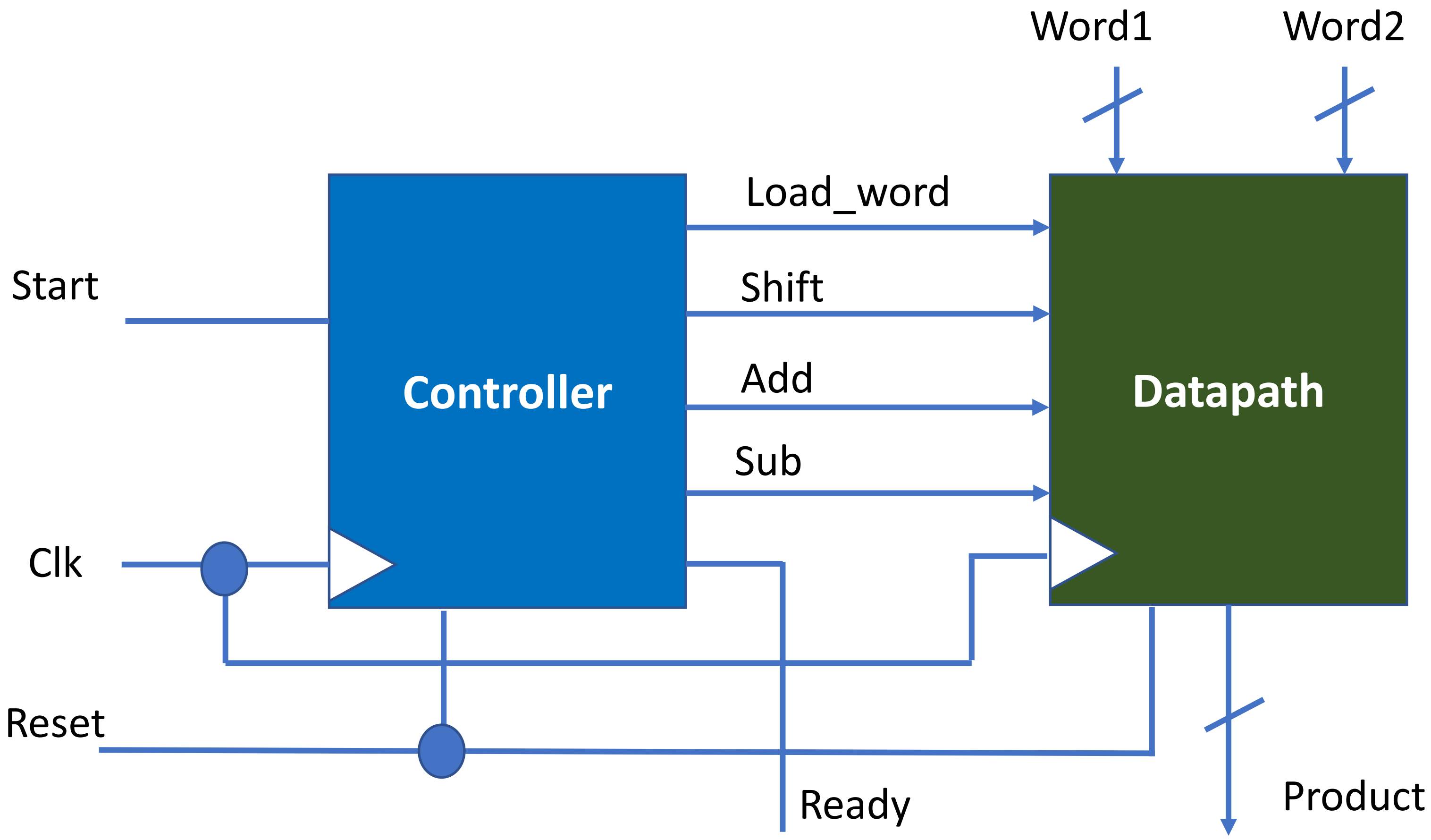
Answer = (0001 0010 0101 1001)2 = (1259)Hex = ($1x16^3 + 2x16^2 + 5x16^1 + 9x16^0$) = 4697_{10}

STG for a 4 Bit Booth Encoded Sequential Multiplier



Machine returns to Idle state after Completion of 4 bit Booth Encoded multiplication

Data Path Architecture of a Booth Sequential Multiplier



Question?

Perform the following multiplication using Booth Encoding.

Multiplicand = 35, Multiplier = 19

How many Adds and Shifts are required in this multiplication?

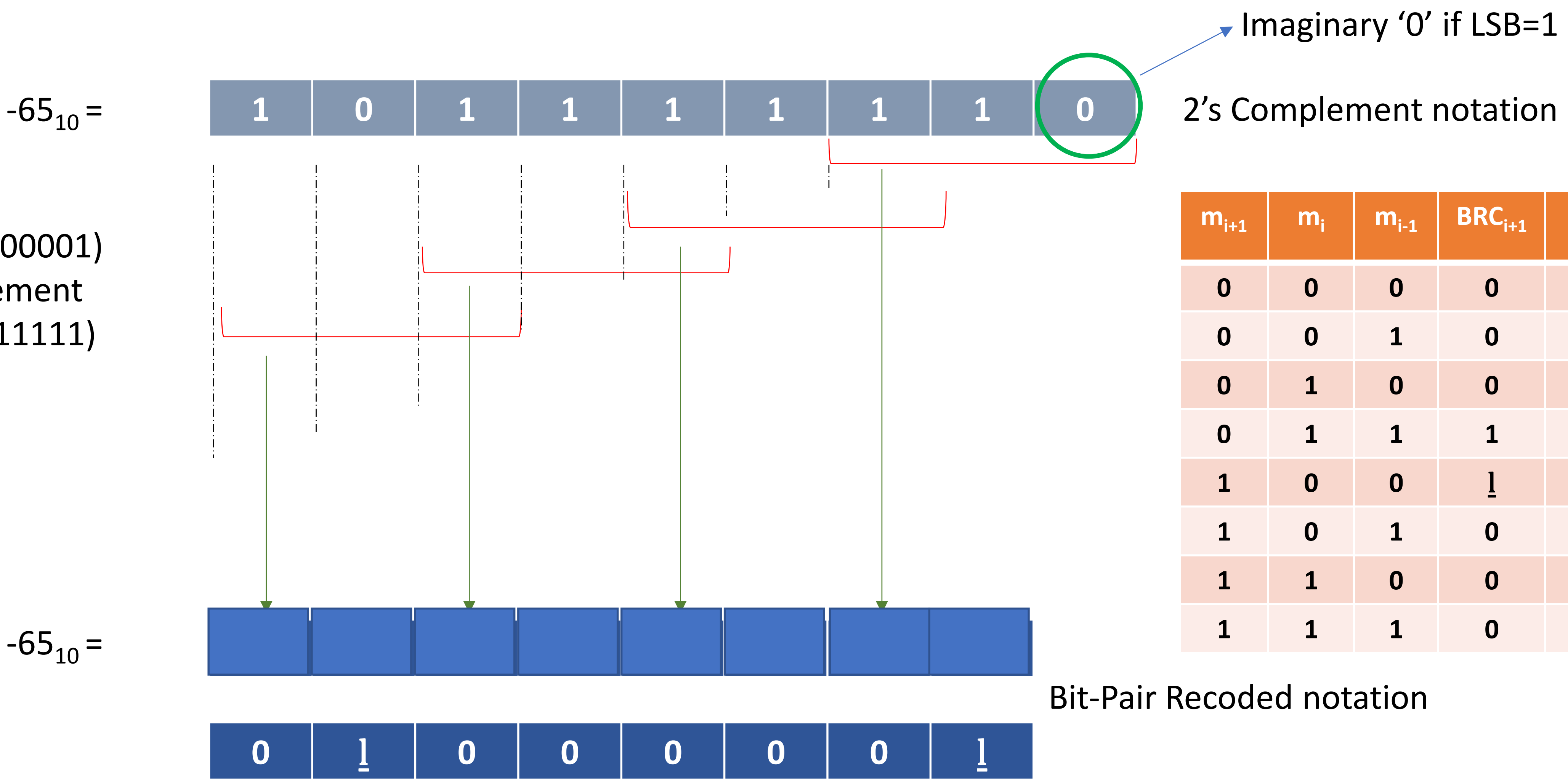
How does this compare to a simple binary array multiplier?

Bit-Pair Encoding
Modified Booth Encoding
Radix-4 Encoding

m_{i+1}	m_i	m_{i-1}	Code	BRC_{i+1}	BRC_i	Value	Status	Multiply Actions
0	0	0	0	0	0	0	String of 0s	Shift by 2
0	0	1	1	0	1	+1	End of string of 1s	Add
0	1	0	2	0	1	+1	Single 1	Add
0	1	1	3	1	0	+2	End of string of 1s	Shift by 1, Add, Shift by 1
1	0	0	4	$\underline{1}$	0	-2	Begin of string of 1s	Shift by 1, Subtract, Shift by 1
1	0	1	5	0	$\underline{1}$	-1	Single 0	Subtract
1	1	0	6	0	$\underline{1}$	-1	Begin of string of 1s	Subtract
1	1	1	7	0	0	0	Midstring of 1s	Shift by 2

Bit-Pair / Radix-4 Recoding of -65_{10}

+65 = (01000001)
2's Complement
-65 = (10111111)



m_{i+1}	m_i	m_{i-1}	BRC_{i+1}	BRC_i	Value
0	0	0	0	0	0
0	0	1	0	1	+1
0	1	0	0	1	+1
0	1	1	1	0	+2
1	0	0	$\underline{1}$	0	-2
1	0	1	0	$\underline{1}$	-1
1	1	0	0	$\underline{1}$	-1
1	1	1	0	0	0

Question of Bit-Pair/Radix-4 Encoding

Express **-75_{10}** in Radix-4 Encoded format using 8 bits to express the given number

m_{i+1}	m_i	m_{i-1}	BRC_{i+1}	BRC_i	Value
0	0	0	0	0	0
0	0	1	0	1	+1
0	1	0	0	1	+1
0	1	1	1	0	+2
1	0	0	$\underline{1}$	0	-2
1	0	1	0	$\underline{1}$	-1
1	1	0	0	$\underline{1}$	-1
1	1	1	0	0	0

$+75_{10}$ = (64+8+2+1)= $(0100\ 1011)_2$

Thus 2's Complement

= $(1011\ 0101)_2$ = -75

1 0 1 1 0 1 0 1[0]

2; coded 01

2; coded 01

6; coded 0 -1

5; coded 0 -1

Radix 4 Encoded = 0 -1 0 -1 0 1 0 1

Radix 4 Encoded = 0 $\underline{1}$ 0 $\underline{1}$ 0 1 0 1

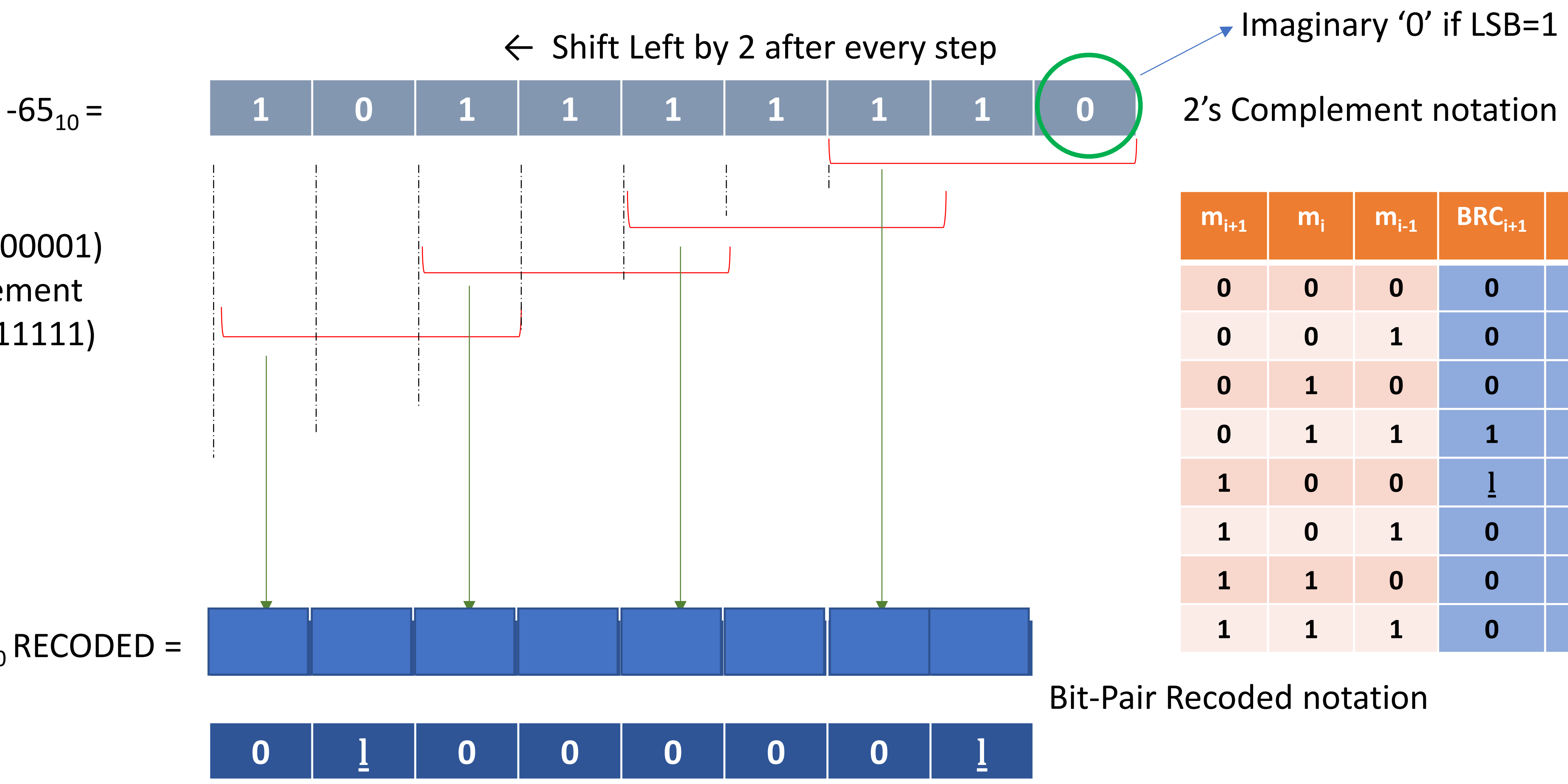
Bit-Pair Encoding
Modified Booth Encoding
Radix-4 Encoding

Shifting by 2 in each step

m_{i+1}	m_i	m_{i-1}	Code	BRC_{i+1}	BRC_i	Value	Status	Multiply Actions
0	0	0	0	0	0	0	String of 0s	Shift Left by 2
0	0	1	1	0	1	+1	End of string of 1s	Add, Shift Left by 2
0	1	0	2	0	1	+1	Single 1	Add, Shift Left by 2
0	1	1	3	1	0	+2	End of string of 1s	Shift by 1, Add, Shift by 1
1	0	0	4	$\underline{1}$	0	-2	Begin of string of 1s	Shift by 1, Subtract, Shift by 1
1	0	1	5	0	$\underline{1}$	-1	Single 0	Subtract, Shift Left by 2
1	1	0	6	0	$\underline{1}$	-1	Begin of string of 1s	Subtract, Shift Left by 2
1	1	1	7	0	0	0	Mid-string of 1s	Shift Left by 2

Bit-Pair / Radix-4 Recoding of -65_{10}

+65 = (01000001)
2's Complement
-65 = (10111111)



m_{i+1}	m_i	m_{i-1}	BRC_{i+1}	BRC_i	Value
0	0	0	0	0	0
0	0	1	0	1	+1
0	1	0	0	1	+1
0	1	1	1	0	+2
1	0	0	$\underline{1}$	0	-2
1	0	1	0	$\underline{1}$	-1
1	1	0	0	$\underline{1}$	-1
1	1	1	0	0	0

Question of Bit-Pair/Radix-4 Encoding

Express **-75_{10}** in Radix-4 Encoded format using 8 bits to express the given number

m_{i+1}	m_i	m_{i-1}	BRC_{i+1}	BRC_i	Value
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**+1**
0	**1**	**0**	**0**	**1**	**+1**
0	**1**	**1**	**1**	**0**	**+2**
1	**0**	**0**	$\underline{1}$	**0**	**-2**
1	**0**	**1**	**0**	$\underline{1}$	**-1**
1	**1**	**0**	**0**	$\underline{1}$	**-1**
1	**1**	**1**	**0**	**0**	**0**

$+75_{10} = (64+8+2+1) = (0100\ 1011)_2$

Thus 2's Complement

$= (1011\ 0101)_2 = -75$

1 0 1 1 0 1 0 1[0]

2; coded 01

2; coded 01

6; coded 0 -1

5; coded 0 -1

Radix 4 Encoded = 0 -1 0 -1 0 1 0 1

Radix 4 Encoded = 0 $\underline{1}$ 0 $\underline{1}$ 0 1 0 1

Radix 4 Coding for Multiplication

m_{i+1}	m_i	m_{i-1}	Code	Multiply Actions
0	0	0	0	Shift Left by 2
0	0	1	1	Add Multiplicand, Shift Left by 2
0	1	0	2	Add Multiplicand, Shift Left by 2
0	1	1	3	Shift by 1, Add Multiplicand, Shift by 1
1	0	0	4	Shift by 1, Subtract Multiplicand, Shift by 1
1	0	1	5	Subtract Multiplicand, Shift Left by 2
1	1	0	6	Subtract Multiplicand, Shift Left by 2
1	1	1	7	Shift Left by 2

Radix 4 Multiplication – Example 1

Imagine Zero bit if LSB = 1

Show Radix 4 Encoded multiplication of 8 x 9, using 8 bits for both numbers

8 = 0000 1000
9 = 0000 1001

Convert 9 = 0000 1001 to Radix 4 Encoded bits

9 = 0 0 0 0 1 0 0 1 [0]

RECODED
010 → 01
100 → -1 0
001 → 01
000 → 00

8 = Multiplicand
X 9 = Recoded Multiplier

									0	**0**	**0**	**0**	**1**	**0**	**0**	**0**
									0	**0**	**0**	**1**	**-1**	**0**	**0**	**1**
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	0	0	0	X	X	X
0	0	0	0	0	0	0	0	0	1	0	0	0	X	X	X	X
												X	X	X	X	X
0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

0 1 = Add Multiplicand, Shl2
-1 0 = Shl 1, Sub, Shl1
0 1 = Add, Shl2
0 0 = Only Shl2, No op

Answer = (0100 1000) = +(64 + 8) = +72_{10}

Radix 4 Multiplication – Example 2

Show Radix 4 Encoded multiplication of 68 x -19, using 8 bits for both numbers

Imagine Zero

-19 = 1 1 1 0 1 1 0 1 [0]

68 = 0100 0100
And 2's Compl is
-68= 1011 1100

19 = 0001 0011
And 2's Compl is
-19= 1110 1101

Convert -19 = 1110 1101 to Radix 4 Encoded bits

RECODED
010 → 01
110 → 0-1
110 → 0-1
111 → 00

68 = Multiplicand										**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	
X -19 = Recoded Multiplier										**0**	**0**	**0**	**-1**	**0**	**-1**	**0**	**1**	
		0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0 1 = Add Multiplicand, Shl2
	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	X	X	0 -1 = Sub, Shl2
	1	1	1	1	1	1	0	1	1	1	1	0	0	X	X	X	X	0 -1 = Sub, Shl2
													X	X	X	X	X	0 0 = Only Shl2, No op
Result	1	1	1	1	1	1	0	1	0	1	1	1	1	0	1	0	0	

Take 2's Complement of Result = -(0101 0000 1100) = -(50C) Hex = $-(1292)_{10}$

Radix 4 Multiplication – Example 3

Show Radix 4 Encoded multiplication of 76 x 55, using 8 bits for both numbers

Imagine Zero

55 = 0 0 1 1 0 1 1 1 [0]

RECODED
110 → 0-1
011 → 10
110 → 0-1
001 → 01

76 = 0100 1100
And 2's Compl is
-76= 1011 0100

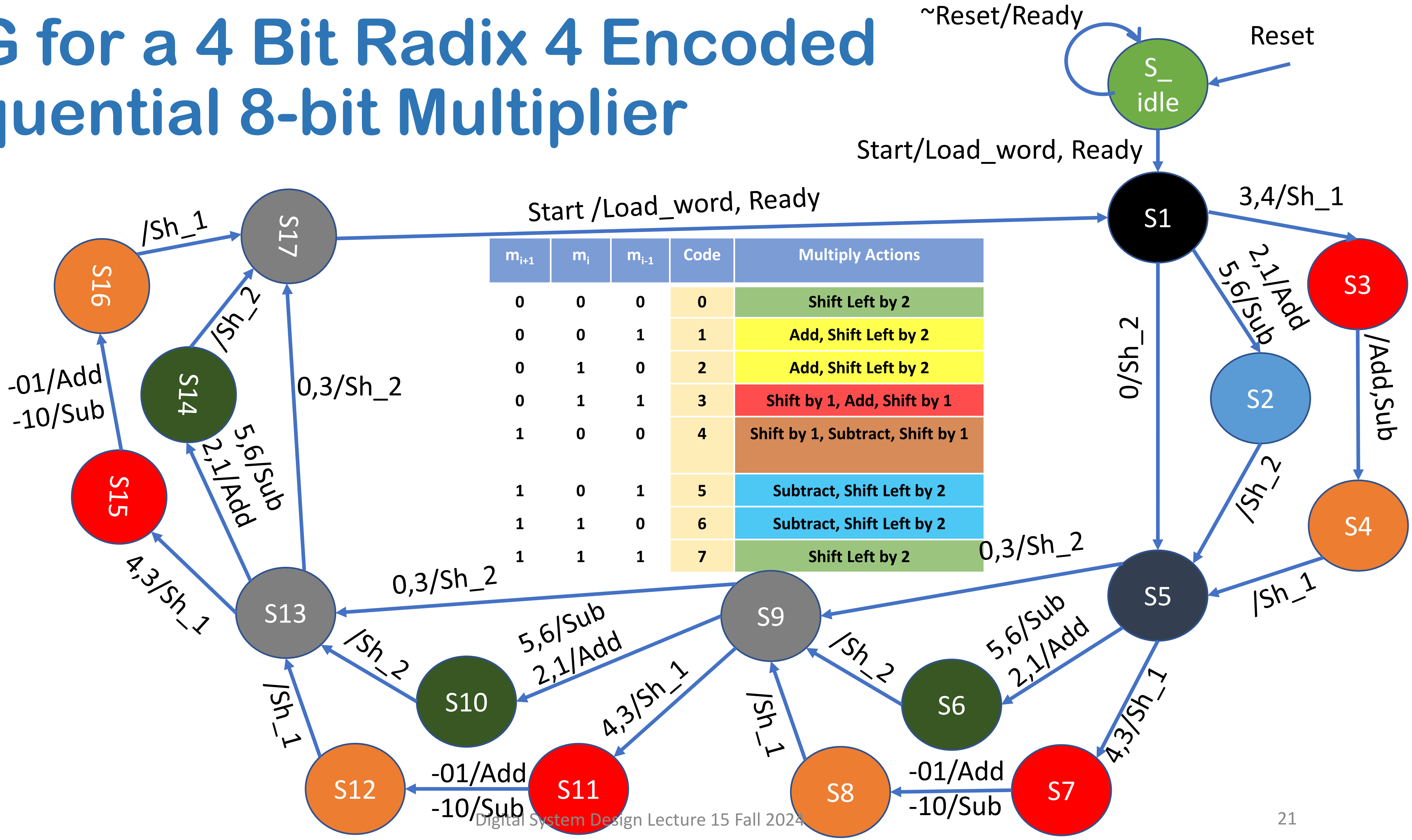
55 = 0011 0111
And 2's Compl is
-55= 1100 1001

Convert 55 = 0011 0111to Radix 4 Encoded bits

76 = Multiplicand										0	1	0	0	1	1	0	0	
X 55 = Recoded Multiplier										0	1	0	-1	1	0	0	-1	
	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	0	0	0 -1 = Sub, Shl2
	0	0	0	0	0	0	0	1	0	0	1	1	0	0	X	X	X	1 0 = Shl1,Add, Shl1
Partial Sum	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	
	1	1	1	1	1	1	0	1	1	0	1	0	0	X	X	X	X	0 -1 = Sub, Shl2
Partial Sum	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	0	0	
	0	0	0	0	1	0	0	1	1	0	0	X	X	X	X	X	X	0 1 = Add, Shl2
Result	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	0	

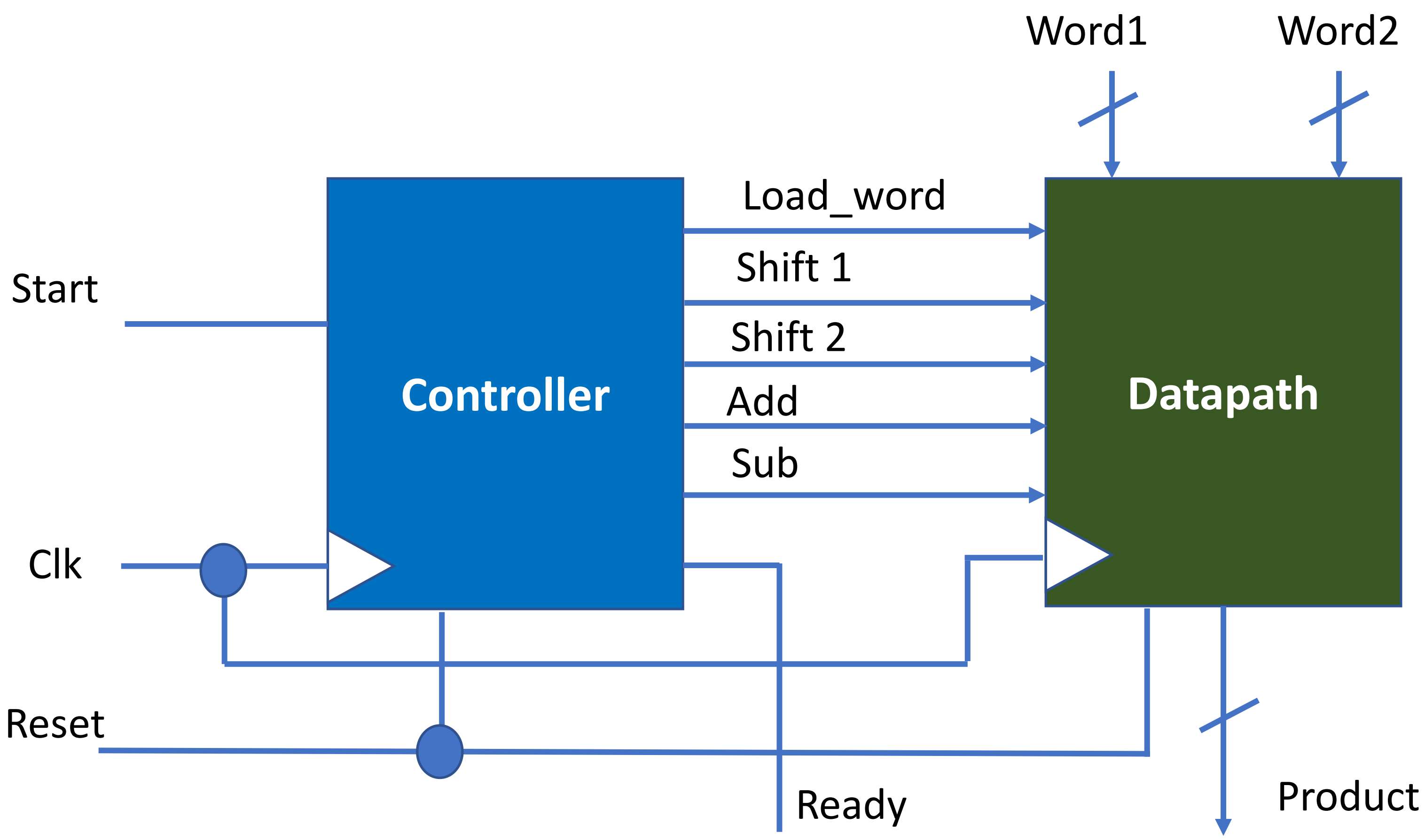
Answer = 0001 0000 0101 0100 = (4+16+64+4096) = $(4180)_{10}$

STG for a 4 Bit Radix 4 Encoded Sequential 8-bit Multiplier



m_{i+1}	m_i	m_{i-1}	Code	Multiply Actions
0	0	0	0	Shift Left by 2
0	0	1	1	Add, Shift Left by 2
0	1	0	2	Add, Shift Left by 2
0	1	1	3	Shift by 1, Add, Shift by 1
1	0	0	4	Shift by 1, Subtract, Shift by 1
1	0	1	5	Subtract, Shift Left by 2
1	1	0	6	Subtract, Shift Left by 2
1	1	1	7	Shift Left by 2

Data Path Architecture of a Radix 4 Sequential Multiplier

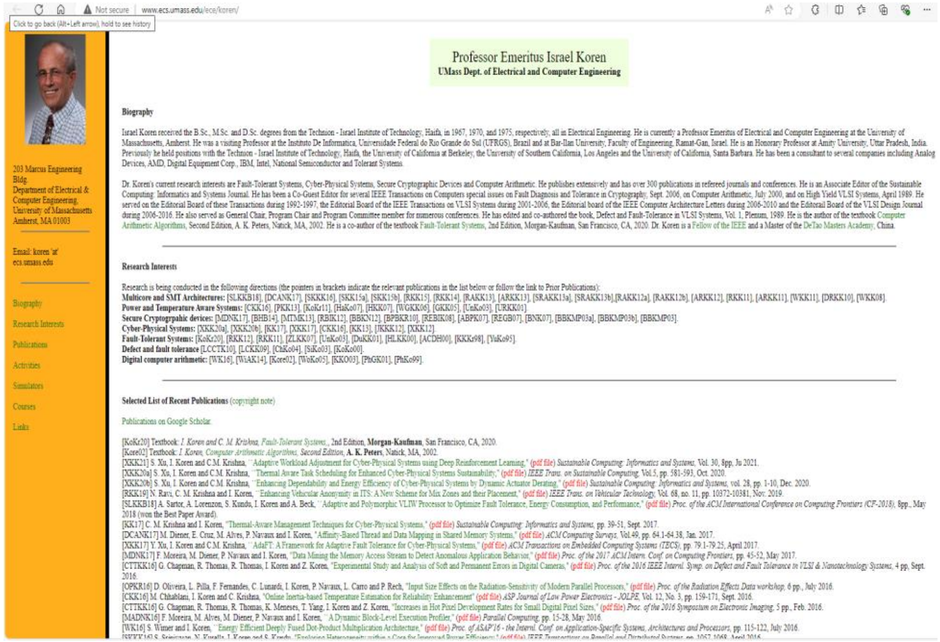


Question?

- Perform the following multiplication using Radix 4 Encoding.
- Multiplicand = 38, Multiplier = 23 (bits allocated?)
- How many Adds and Shifts are required in this multiplication?
- How does this compare to a simple binary array multiplier?

Online Booth Encoding Simulator

http://www.ecs.umass.edu/ece/koren/arith/simulator/Booth/



Prof. Korean Page:

http://www.ecs.umass.edu/ece/koren/

Simulators Tab is on the left

Many simulators of Computer Arithmetic are available:

http://www.ecs.umass.edu/ece/koren/arith/simulator/

Online Modified Booth (Radix 4) Encoding

http://www.ecs.umass.edu/ece/koren/arith/simulator/ModBooth/

Simulator available on Prof Koren's website

Delay computation in binary array multiplier

Previous topic:

Delay computation in Array Multiplier (binary inputs):

http://www.ecs.umass.edu/ece/koren/arith/simulator/ArrMlt/