Lecture 16

EE 421 / CS 425

Digital System Design

Spring 2023

Shahid Masud

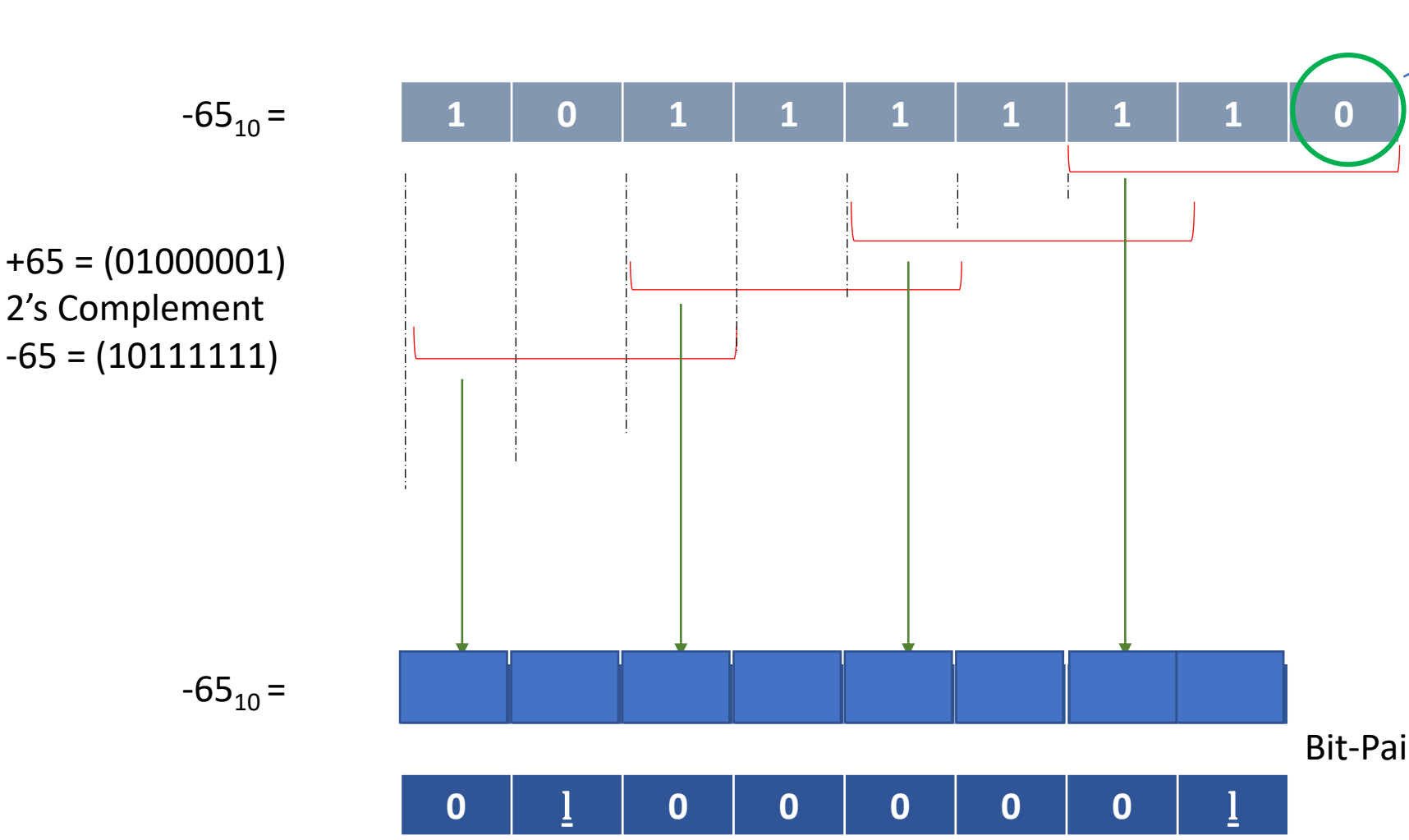
Topics

- Examples: Booth / Radix 4 Multiplication
- Binary Divider Operation
- Binary Divider Circuit
- STG of Divider
- Floating Point Representation (if time permits)
- Floating Point Multiplier – design and operation

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, Shift by 2
0	1	0	2	0	1	+1	Single 1	Add, Shift 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 by 2
1	1	0	6	0	$\underline{1}$	-1	Begin of string of 1s	Subtract, Shift by 2
1	1	1	7	0	0	0	Midstring of 1s	Shift by 2

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



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	1̲	0	-2
1	0	1	0	1̲	-1
1	1	0	0	1̲	-1
1	1	1	0	0	0

Question of Bit-Pair/Radix-4 Encoding

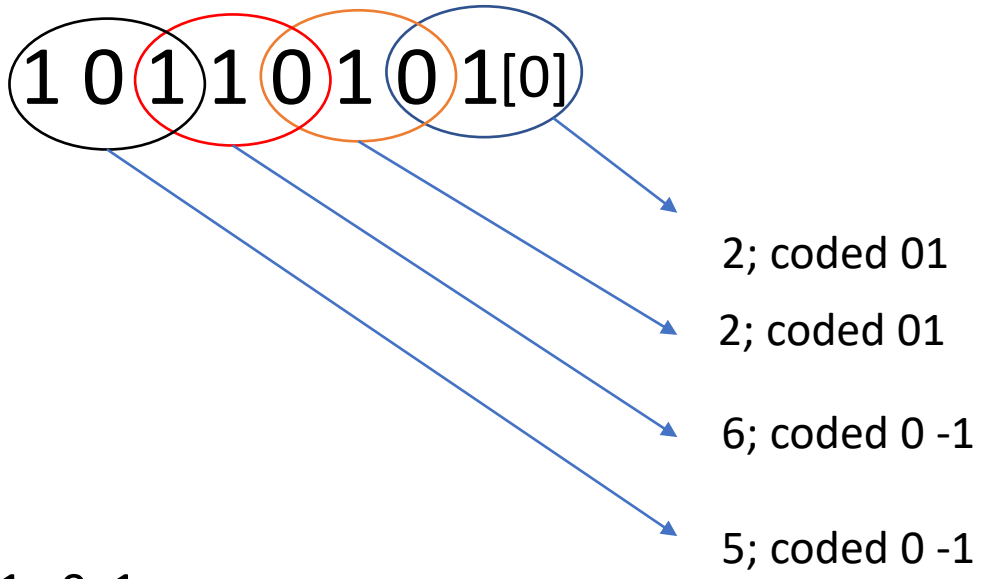
Express **-75$_{10}$** in Radix-4 Encoded format using 8 bits to express the given number

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

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



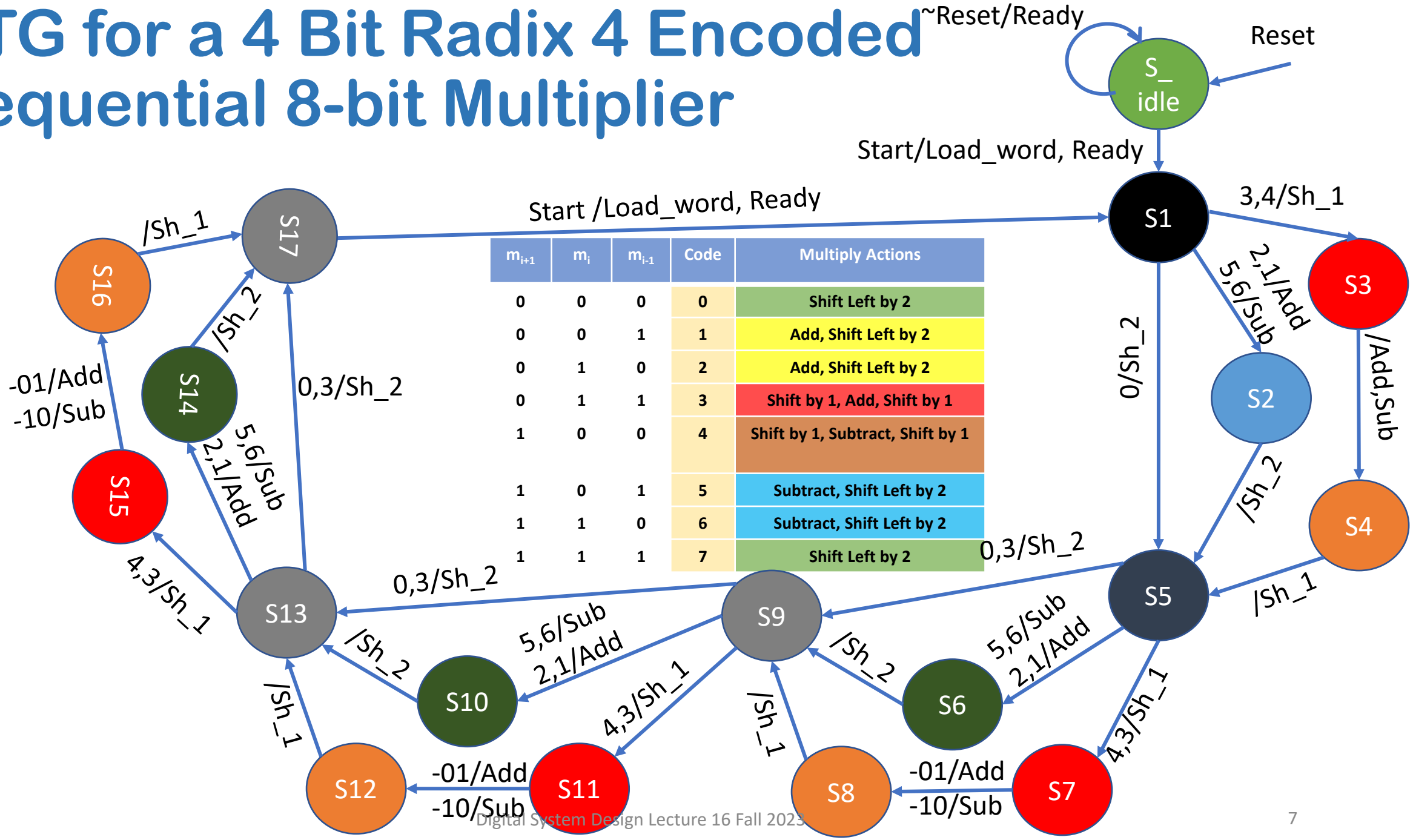
Radix 4 Encoded = 0 -1 0 -1 0 1 0 1

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

Radix 4 Coding for Multiplication

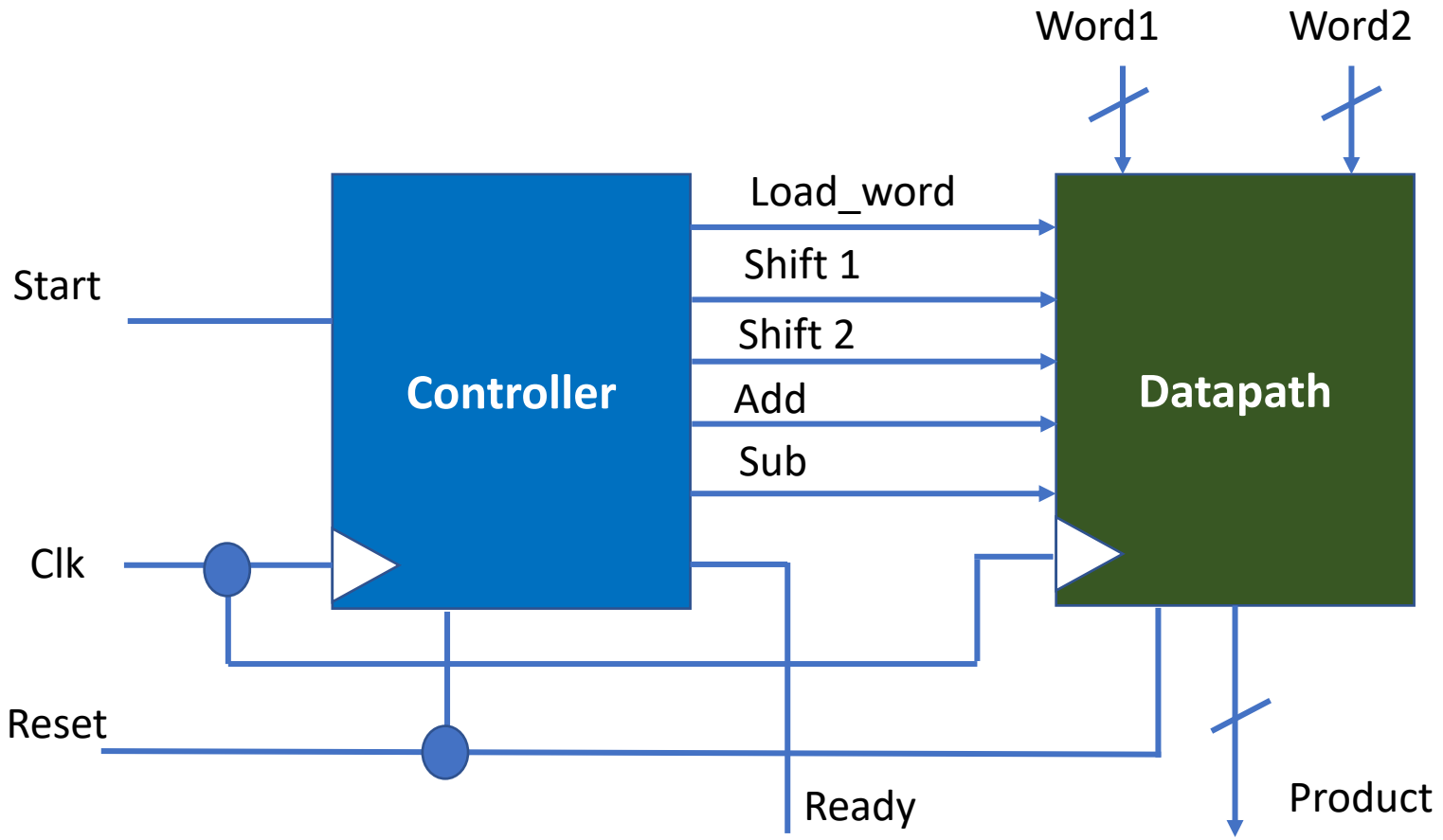
m_{i+1}	m_i	m_{i-1}	Code	Multiply Actions
0	0	0	0	Shift Left by 2
0	0	1	1	Add Multiplicand, Shift Left by 2
0	1	0	2	Add Multiplicand, Shift Left by 2
0	1	1	3	Shift by 1, Add Multiplicand, Shift by 1
1	0	0	4	Shift by 1, Subtract Multiplicand, Shift by 1
1	0	1	5	Subtract Multiplicand, Shift Left by 2
1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

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



Radix 4 Multiplication – Example 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

Imagine Zero bit if LSB = 1

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	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]

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

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

68 = Multiplicand
X -19 = Recoded Multiplier

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

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

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

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

Imagine Zero

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

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

76 = Multiplicand
X 55 = Recoded Multiplier

										0	**1**	**0**	**0**	**1**	**1**	**0**	**0**
										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	0	0	0	0	0	0	1	0	0	1	1	0	0	X	X	X
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
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
Result	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	0

0 -1 = Sub, Shl2

1 0 = Shl1,Add, Shl1

0 -1 = Sub, Shl2

0 1 = Add, Shl2

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

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?

Division Operation in Decimal Numbers

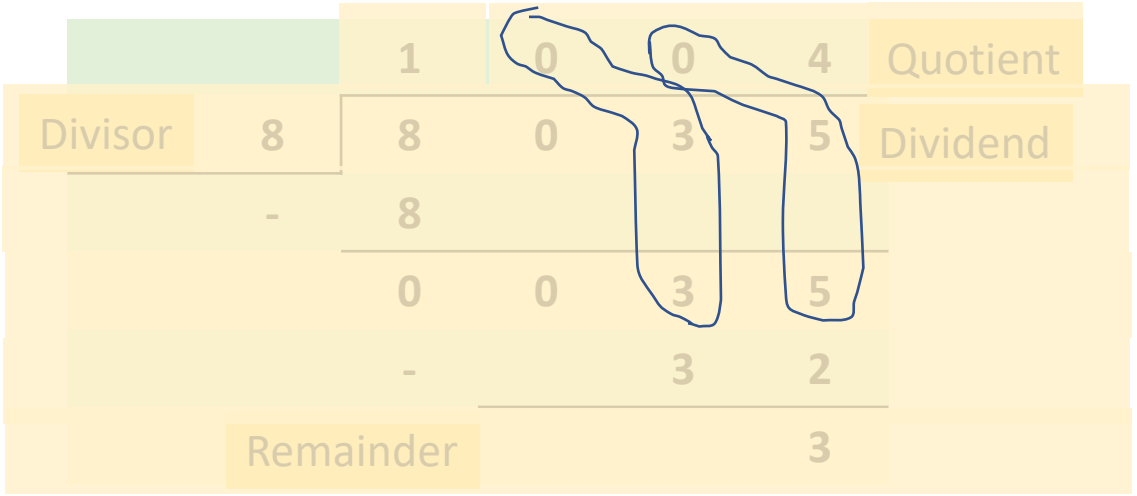
Division of 274 ÷ 13

				2	1	Quotient
Divisor	1	3	2	7	4	Dividend
		-	-2	6		
				1	4	
			-	1	3	
		Remainder			1	Rem

Division Operation in Decimal Numbers

Division of 299 ÷ 15

Decimal Division – another example

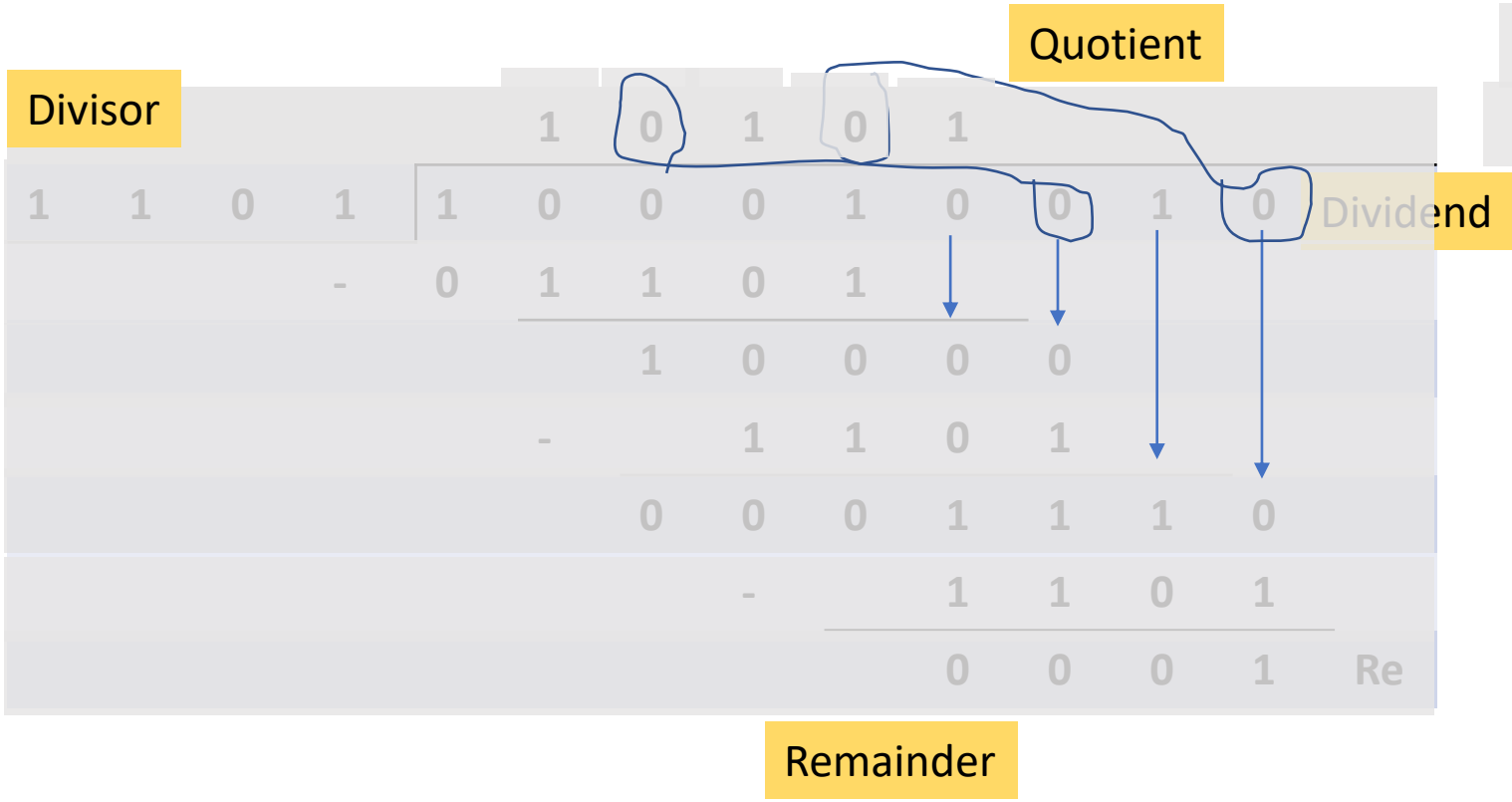


Division Operation in Binary – Example 1

Remainder

Division of 274 ÷ 13

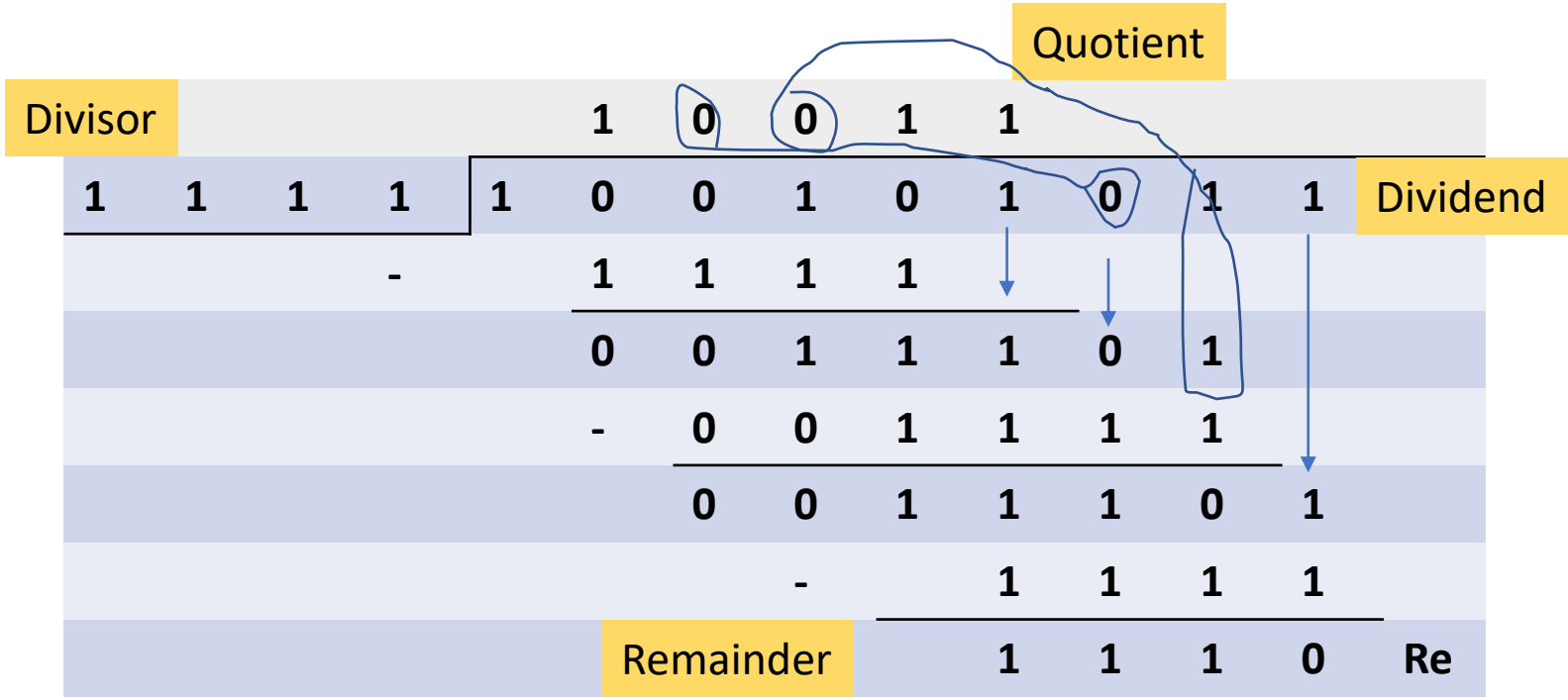
			2	1	
1	3	2	7	4	
	-	-2	6		
			1	4	
		-	1	3	
				1	Rem



Division Operation in Binary – Example 2

Division of 299 ÷ 15

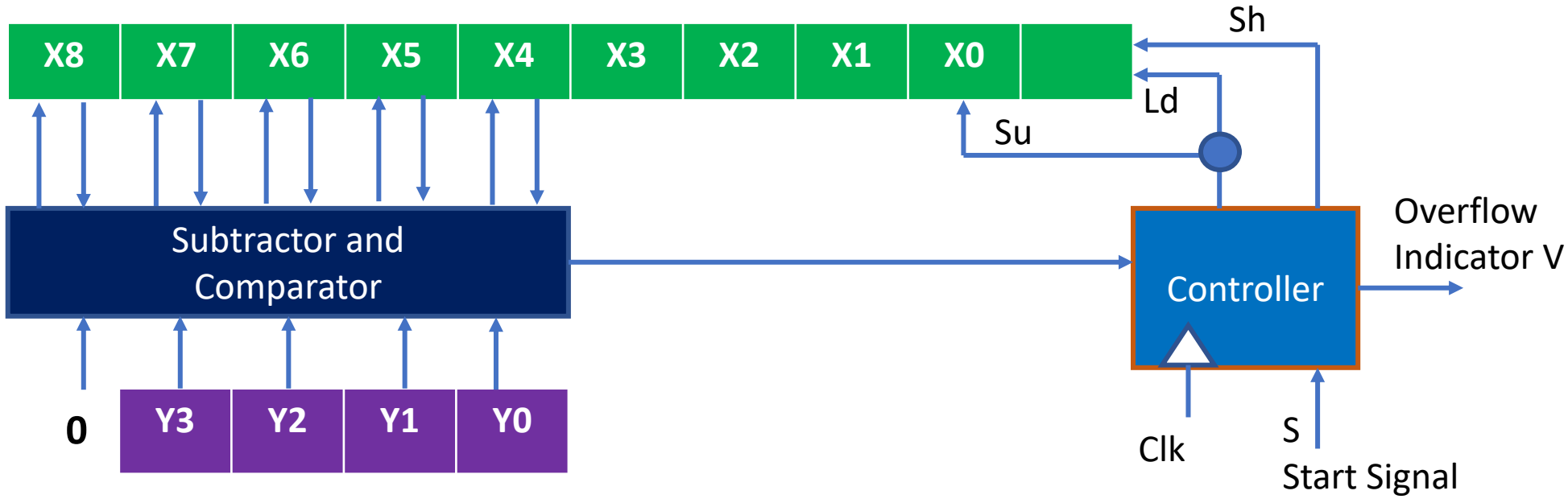
```
          1   9
    ------------
1   5 | 2   9   9
      -   2   8   5
    ------------
          1   4     Rem
```



Division Operation in Binary – Example 3

						1	1	0	1	Quotient			
Divisor	1	0	1	1	1	0	0	1	0	0	1	1	Dividend
				-		1	0	1	1				
						0	1	1	1	0			
				-			1	0	1	1			
							0	0	1	1	1	1	
					-				1	0	1	1	
					Remainder				0	1	0	0	Re

Block Diagram of Sequential Binary Divider



Dividend Register

0	1	0	0	0	0	1	1	1

Divisor Register

1	1	0	1

Overflow V = As a result of a divison operation, if the quotient requires more bits than are available for storing quotient

Operation of Sequential Binary Divider

Show binary division
135 ÷ 13

```
                1 0 1 0      Growing bits
1 1 0 1 | 1 0 0 0 0 1 1 1    Reducing bits
        -   1 1 0 1
            0 1 1 1
        -   0 0 0 0
              1 1 1 1
        -     1 1 0 1
              0 1 0 1
        -     0 0 0 0
                1 0 1
```

Remainder

Dividend

Dividing line between Dividend and Quotient

Divisor

After the shift, the right most position in dividend register is 'empty'

Subtraction is now carried out. The first quotient digit of 1 is stored in the unused portion of the dividend register

First quotient digit

Next we shift the dividend one place to the left

Since subtraction yields negative result so we shift dividend to the left again, and the second quotient bit remains 0

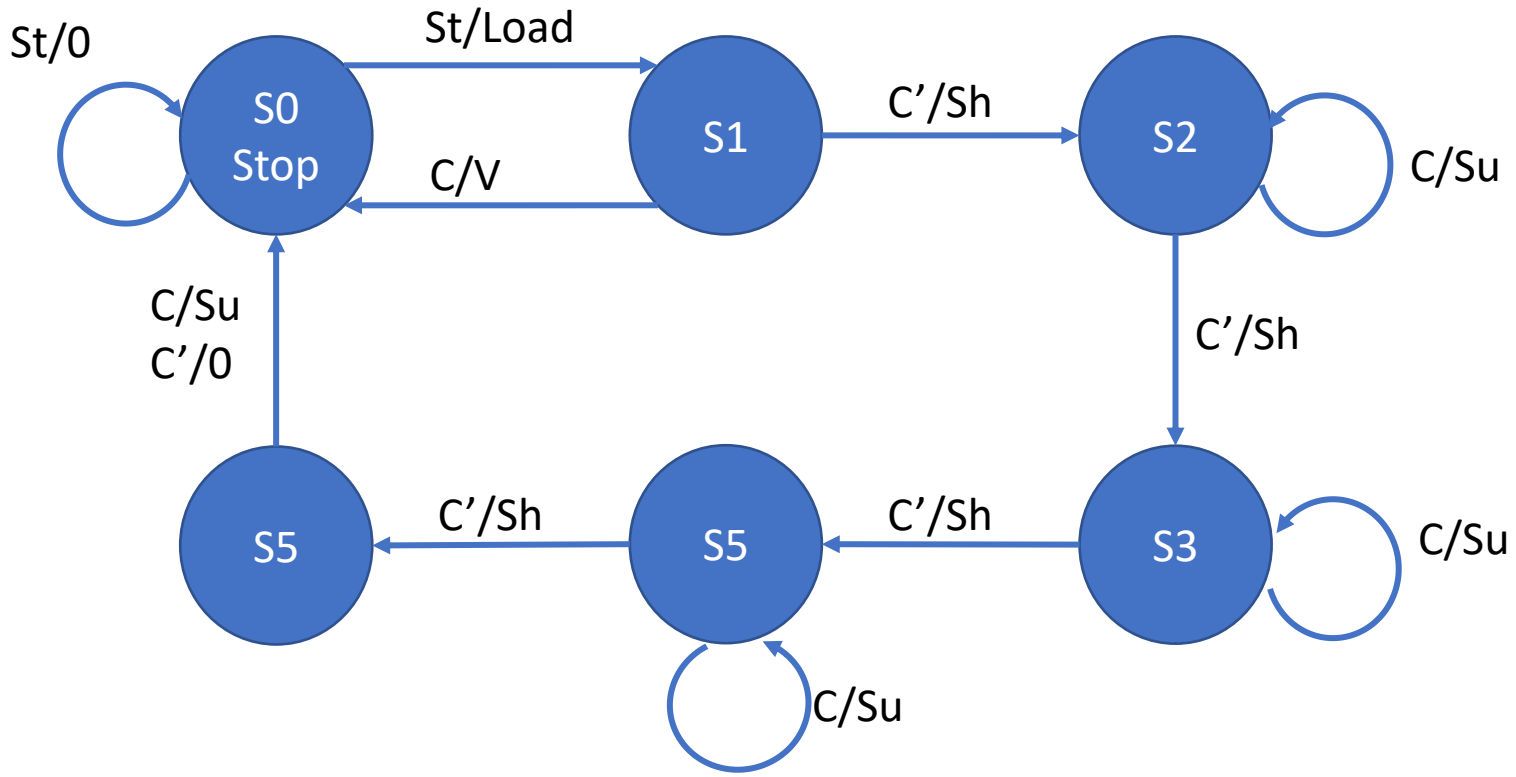
Subtraction is now carried out, the third quotient digit of 1 is stored in the unused portion of the dividend register

Third quotient digit

A final shift is done and fourth quotient bit is set to 0

Quotient

STG of a Binary Divider



Su = Subtract Signal
C = Comparator Output
If divisor is greater than 5 leftmost dividend bits (as per given number), then C=0; otherwise C=1
Whenever C=1, then subtract signal is generated and quotient bit is set to 1
Whenever C=0, then subtraction cannot occur without a negative result so a Shift signal Sh is generated

Division Examples

- Try using 2's Complement Add instead of Sub in Division operations