Lecture 15

EE 421 / CS 425

Digital System Design

Fall 2025

Shahid Masud

Topics

Quiz 3 Today

Midterm on Wednesday 23 October

Booth Encoding and Booth Multiplication - Recap

Modified Booth / Radix 4 Conversion

Booth and Radix 4 Multiplication Process

Booth and Radix 4 Multiplication Examples

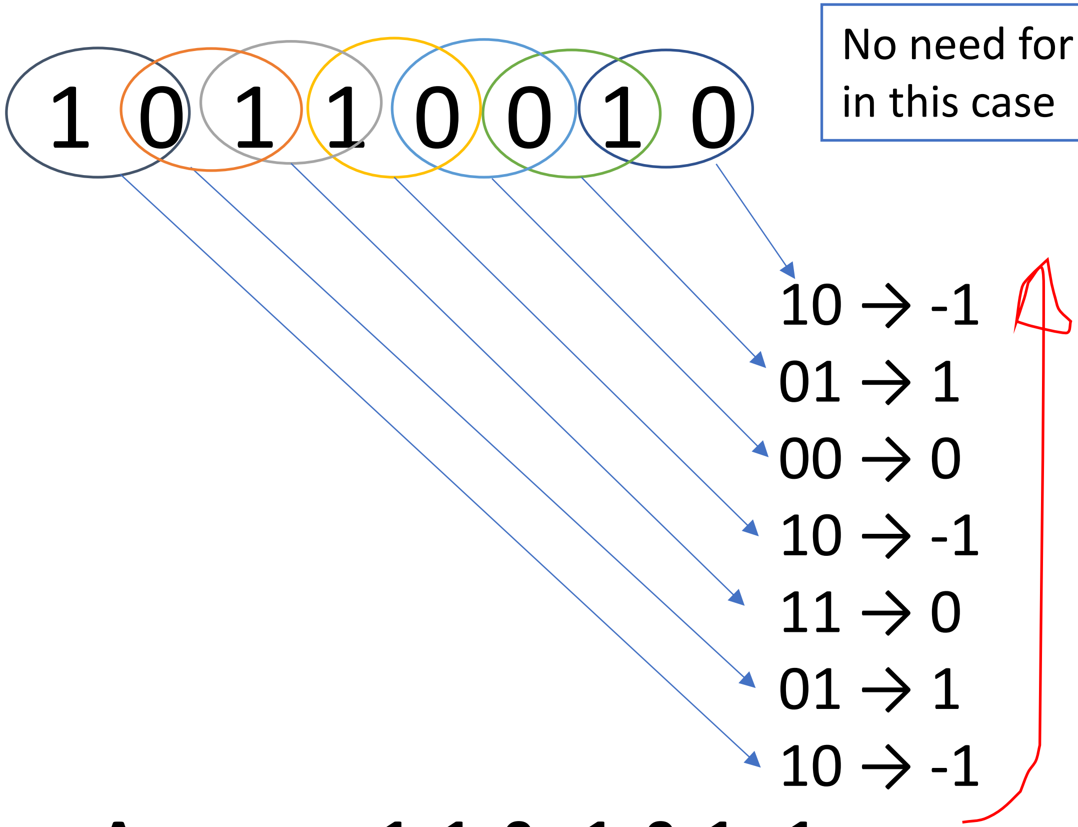
STG for Booth and Radix 4 Sequential Multipliers

Question?

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

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

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



No need for extra ‘0’ after LSB in this case

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	
			0	0	0	0	0	1	1	0	X	
		1	1	1	1	1	0	1	0	X	X	
		0	0	0	0	1	1	0	X	X	X	
								X	X	X	X	
		0	1	0	0	0	1	1	1	1	0	

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

01 = Add Multiplicand to Acc

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

01 = Add Multiplicand to Acc

00 = No Op, Shift Left by 1

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}

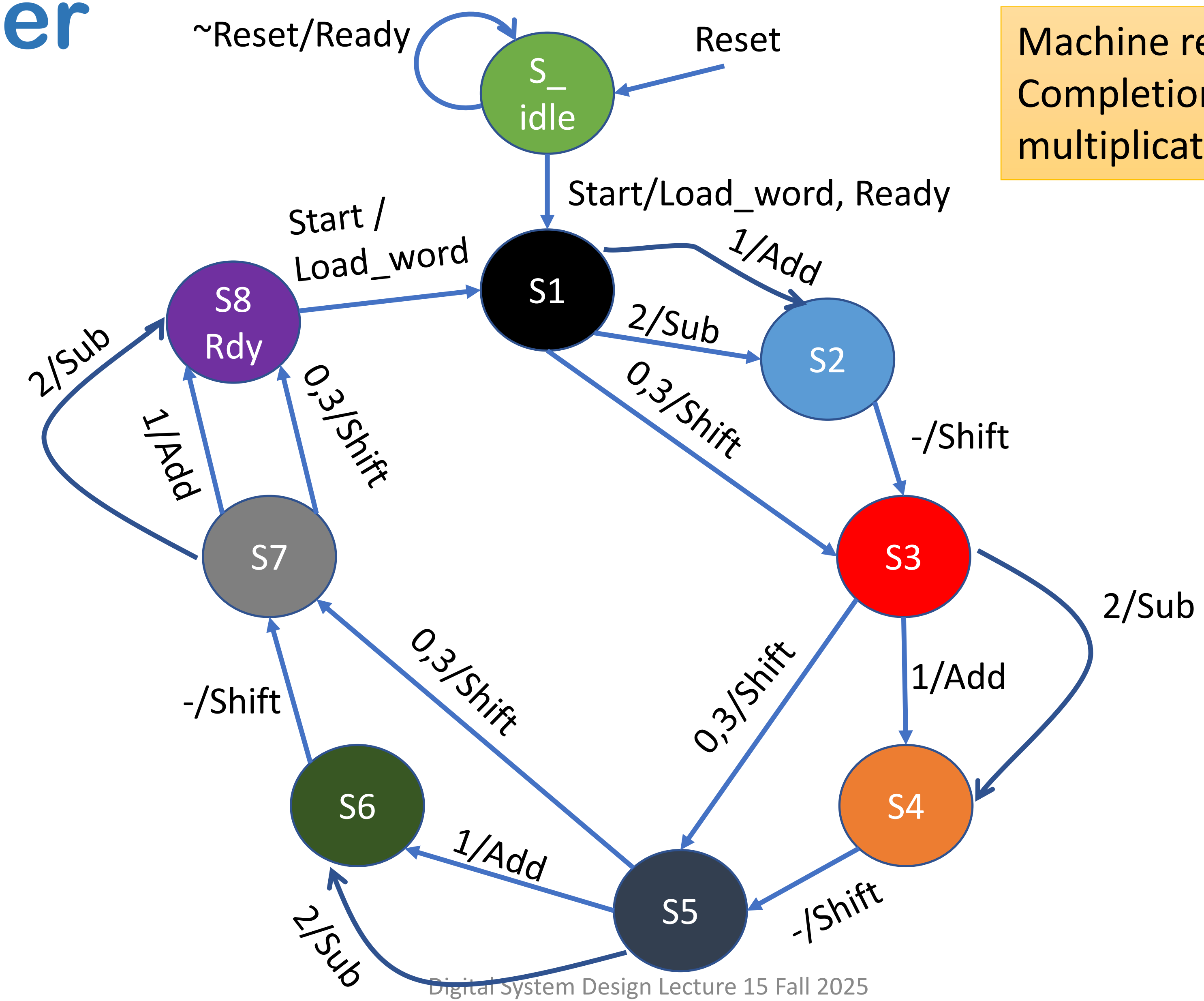
Imaginary Zero bit if LSB = 1

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

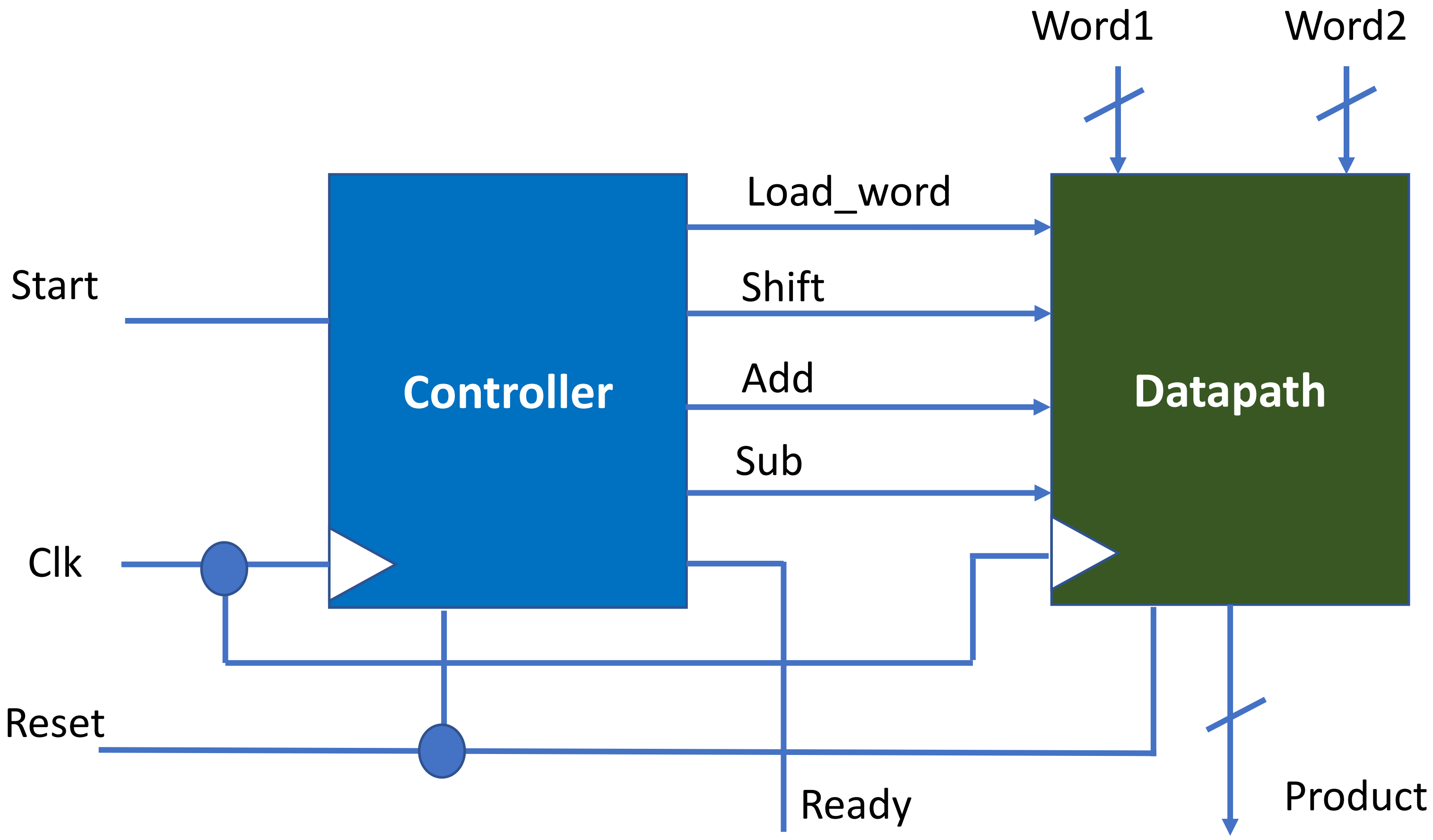
Answer = (0001 0010 0101 1001)2 = (1259)Hex = $(1x16^3 + 2x16^2 + 5x16^1 + 9x16^0)$ = 4697_{10}

STG for a 4 Bit Booth Encoded Sequential Multiplier

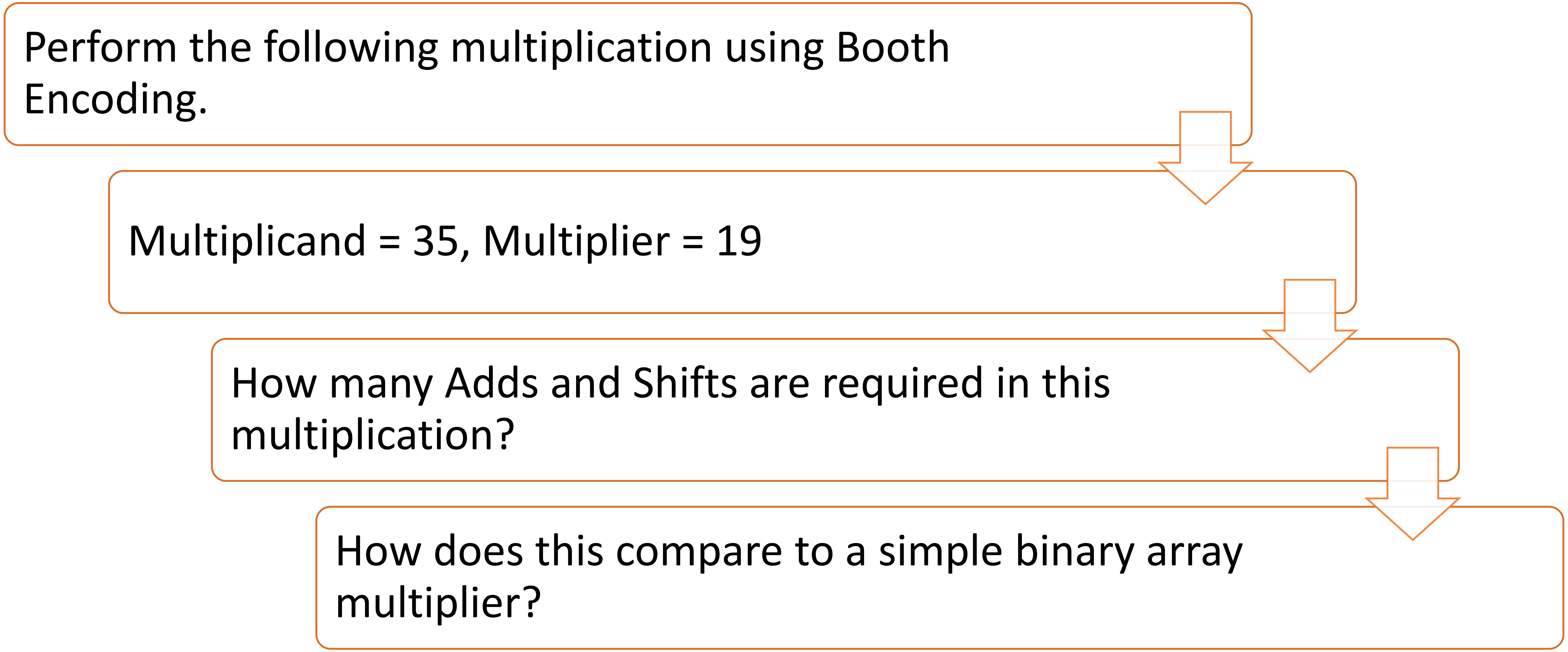


Machine returns to Idle state after Completion of 4 bit Booth Encoded multiplication

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

Check Three Bits

Coded Two Bits

m_{i+1}	m_i	m_{i-1}	Decimal	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}

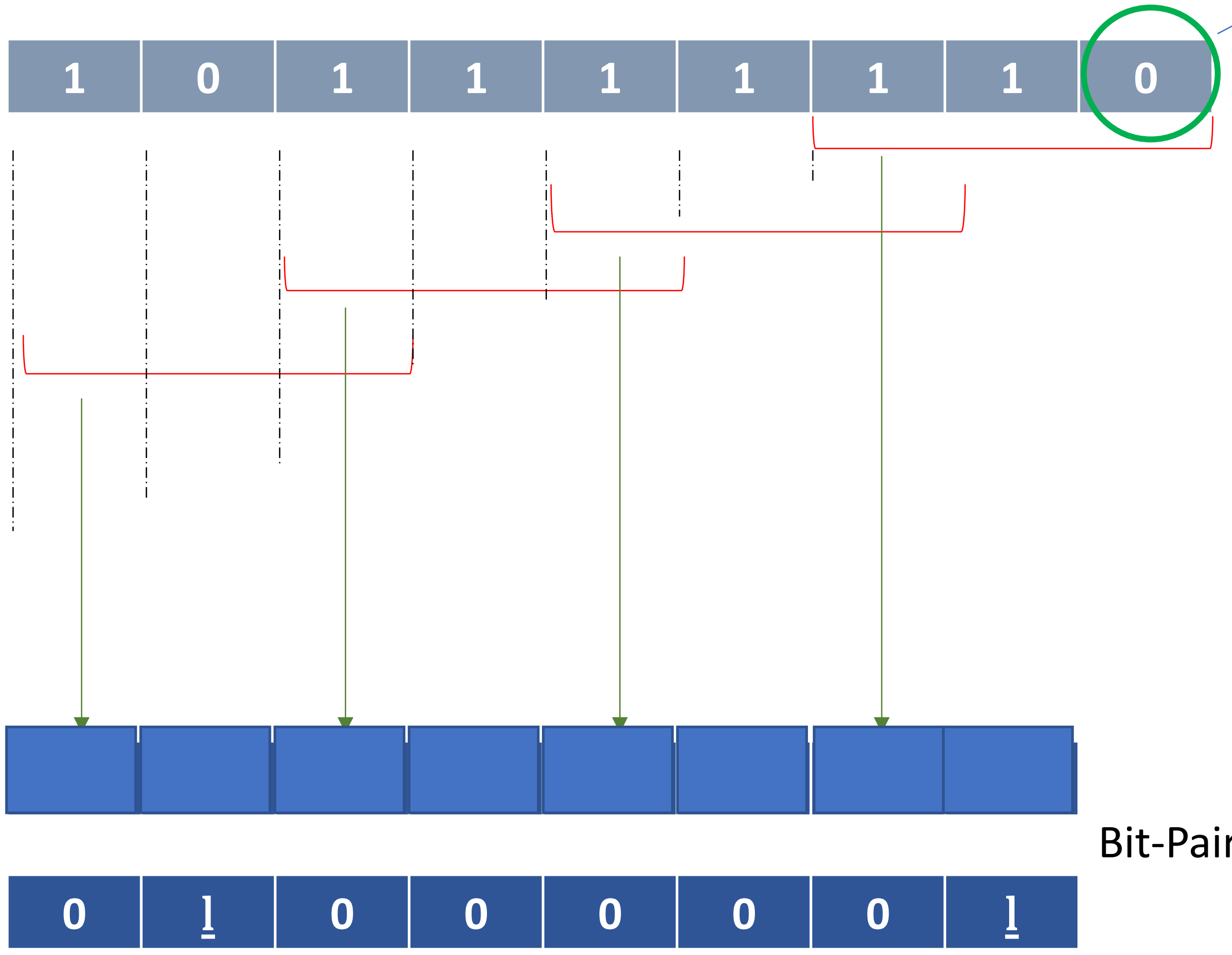
Imaginary ‘0’ if LSB=1

$-65_{10} =$

1	0	1	1	1	1	1	1	0

2’s Complement notation

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



$-65_{10} =$

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	$\underline{1}$	0	-2
1	0	1	0	$\underline{1}$	-1
1	1	0	0	$\underline{1}$	-1
1	1	1	0	0	0

Bit-Pair Recoded notation

0	$\underline{1}$	0	0	0	0	0	$\underline{1}$

Question of Bit-Pair/Radix-4 Encoding

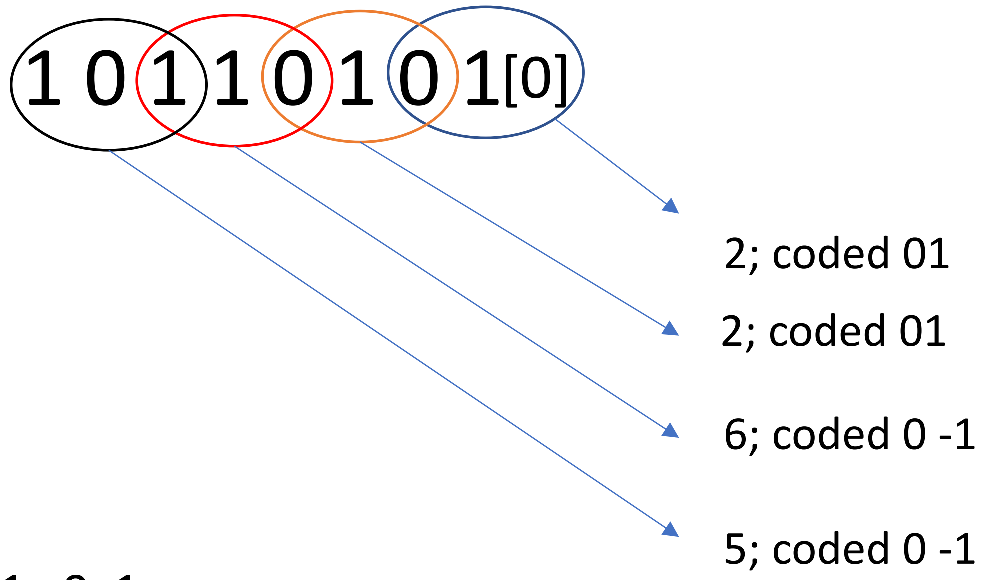
Express **-75**$_{10}$ in Radix-4 Encoded format using 8 bits to express the given number

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

Bit-Pair Encoding
Modified Booth Encoding
Radix-4 Encoding

Shifting by 2 in each step

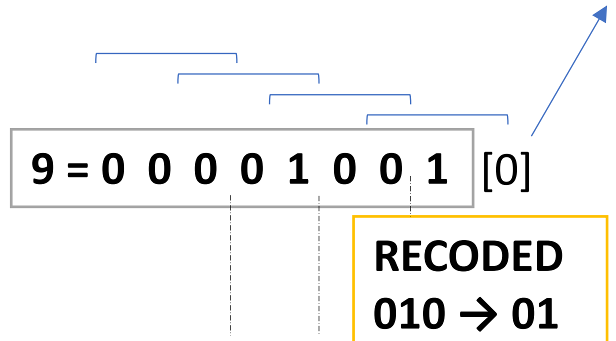
m_{i+1}	m_i	m_{i-1}	Code	BRC_{i+1}	BRC_i	Value	Status	Multiply Actions
0	0	0	0	0	0	0	String of 0s	Shift Left by 2
0	0	1	1	0	1	+1	End of string of 1s	Add, Shift Left by 2
0	1	0	2	0	1	+1	Single 1	Add, Shift Left by 2
0	1	1	3	1	0	+2	End of string of 1s	Shift by 1, Add, Shift by 1
1	0	0	4	$\underline{\text{l}}$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

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

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

8 = Multiplicand										**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	
X 9 = Recoded Multiplier										**0**	**0**	**0**	**1**	**-1**	**0**	**0**	**1**	
		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0 1 = Add Multiplicand, Shl2
	1	1	1	1	1	1	1	1	1	1	1	0	0	0	X	X	X	-1 0 = Shl 1, Sub, Shl1
	0	0	0	0	0	0	0	0	0	1	0	0	0	X	X	X	X	0 1 = Add, Shl2
													X	X	X	X	X	0 0 = Only Shl2, No op
	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	

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

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

Imagine Zero

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

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	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

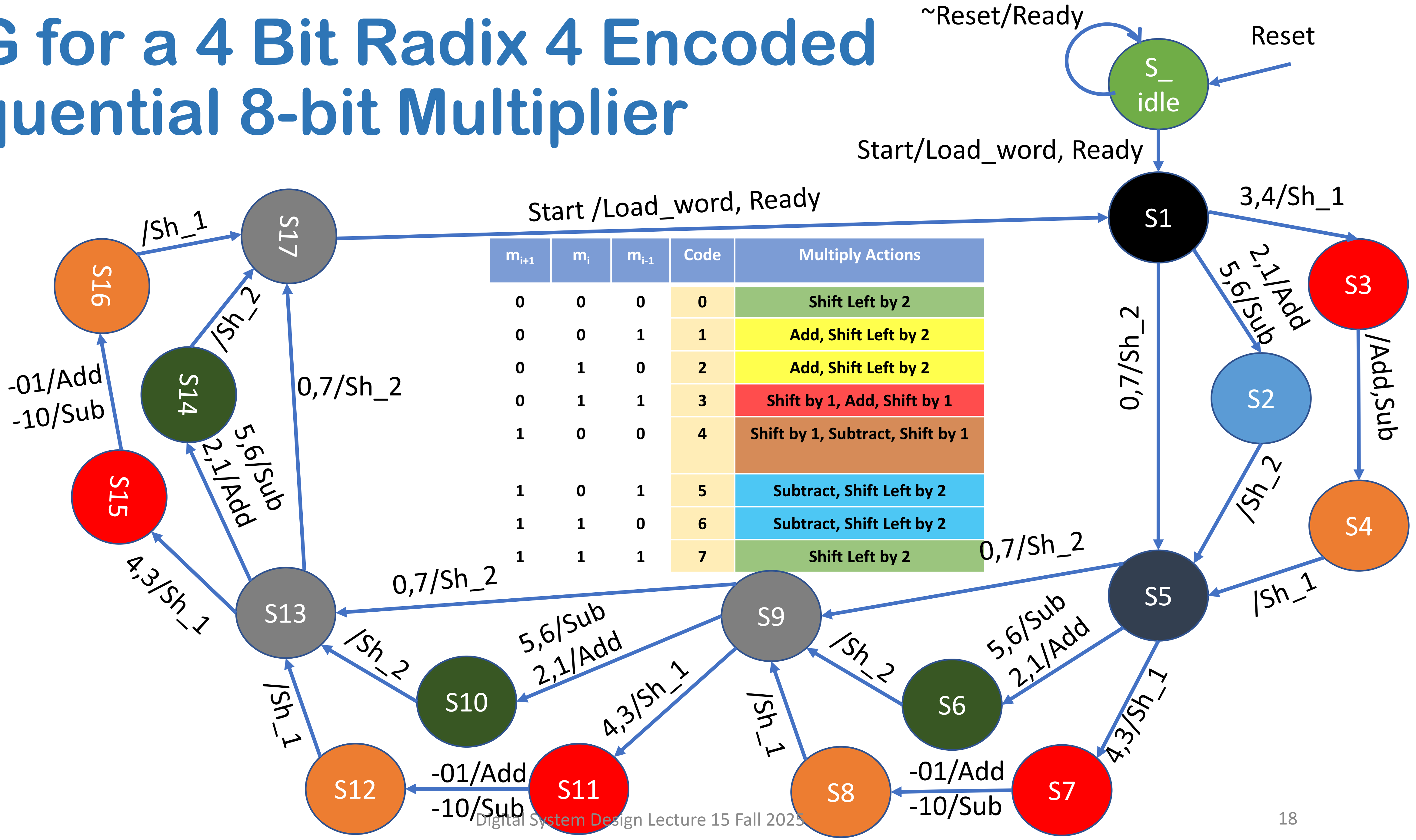
55 = 0011 0111
And 2's Compl is
-55= 1100 1001

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	

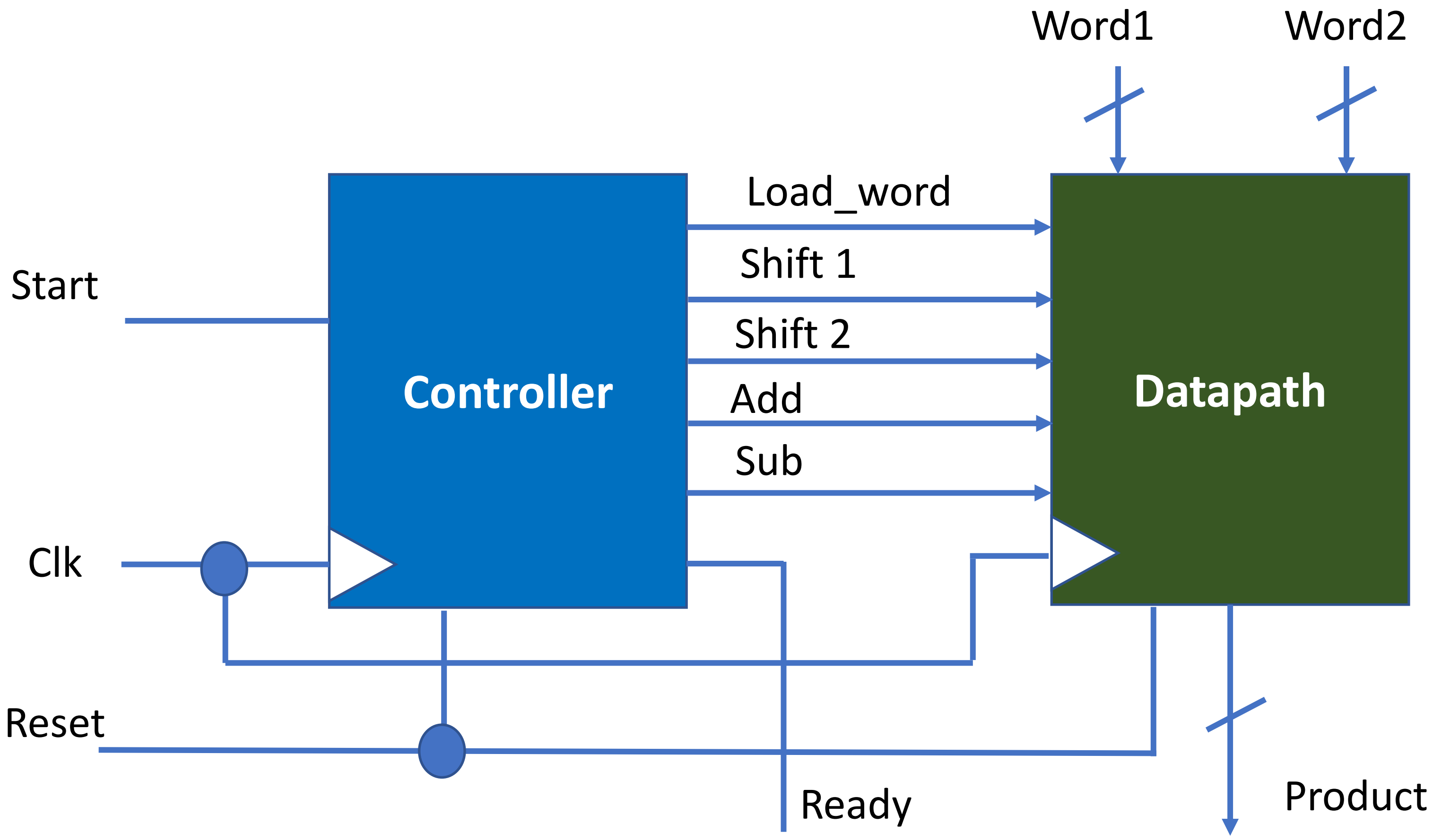
Answer = 0001 0000 0101 0100 = (4+16+64+4096) = $(4180)_{10}$

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.

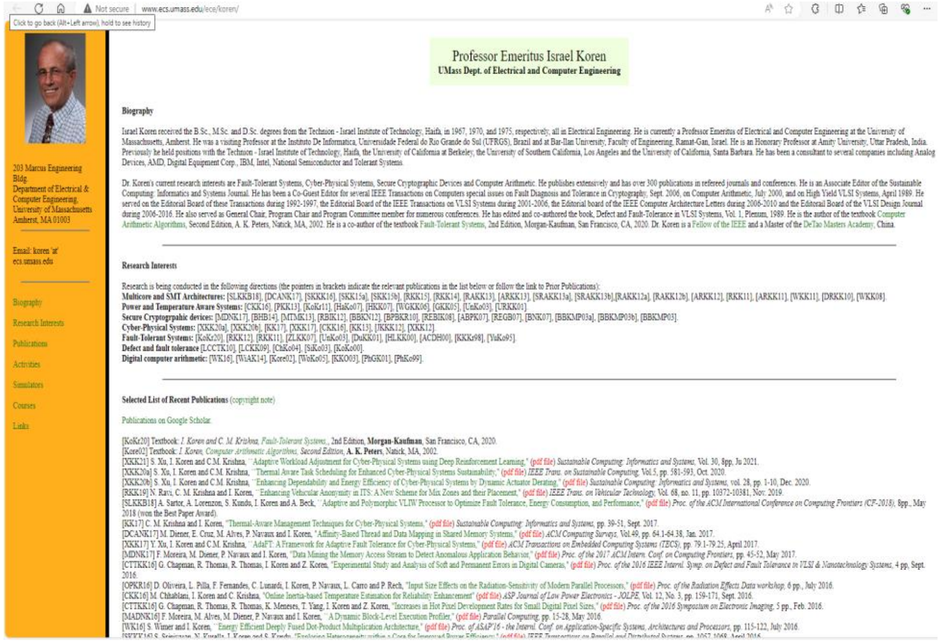
Multiplicand = 38, Multiplier = 23 (bits allocated?)

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/


Not secure
www.ecs.umass.edu/ece/koren/
Click to go back (Alt+Left arrow), hold to see history

Professor Emeritus Israel Koren
UMass Dept. of Electrical and Computer Engineering

203 Marcus Engineering Bldg.
Department of Electrical & Computer Engineering,
University of Massachusetts
Amherst, MA 01003

Email: koren 'at' ecs.umass.edu

Biography
Research Interests
Publications
Activities
Simulators
Courses
Links

Biography

Israel Koren received the B.Sc., M.Sc. and D.Sc. degrees from the Technion - Israel Institute of Technology, Haifa, in 1967, 1970, and 1975, respectively, all in Electrical Engineering. He is currently a Professor Emeritus of Electrical and Computer Engineering at the University of Massachusetts, Amherst. He was a visiting Professor at the Instituto De Informatica, Universidade Federal do Rio Grande do Sul (UFRGS), Brazil and at Bar-Ilan University, Faculty of Engineering, Ramat-Gan, Israel. He is an Honorary Professor at Amity University, Uttar Pradesh, India. Previously he held positions with the Technion - Israel Institute of Technology, Haifa, the University of California at Berkeley, the University of Southern California, Los Angeles and the University of California, Santa Barbara. He has been a consultant to several companies including Analog Devices, AMD, Digital Equipment Corp., IBM, Intel, National Semiconductor and Tolerant Systems.

Dr. Koren's current research interests are Fault-Tolerant Systems, Cyber-Physical Systems, Secure Cryptographic Devices and Computer Arithmetic. He publishes extensively and has over 300 publications in refereed journals and conferences. He is an Associate Editor of the Sustainable Computing: Informatics and Systems Journal. He has been a Co-Guest Editor for several IEEE Transactions on Computers special issues on Fault Diagnosis and Tolerance in Cryptography, Sept. 2006, on Computer Arithmetic, July 2000, and on High Yield VLSI Systems, April 1989. He served on the Editorial Board of these Transactions during 1992-1997, the Editorial Board of the IEEE Transactions on VLSI Systems during 2001-2006, the Editorial board of the IEEE Computer Architecture Letters during 2006-2010 and the Editorail Board of the VLSI Design Journal during 2006-2016. He also served as General Chair, Program Chair and Program Committee member for numerous conferences. He has edited and co-authored the book, Defect and Fault-Tolerance in VLSI Systems, Vol. 1, Plenum, 1989. He is the author of the textbook Computer Arithmetic Algorithms, Second Edition, A. K. Peters, Natick, MA, 2002. He is a co-author of the textbook Fault-Tolerant Systems, 2nd Edition, Morgan-Kaufman, San Francisco, CA, 2020. Dr. Koren is a Fellow of the IEEE and a Master of the DeTao Masters Academy, China.

Research Interests

Research is being conducted in the following directions (the pointers in brackets indicate the relevant publications in the list below or follow the link to Prior Publications):
Multicore and SMT Architectures: [SLKKB18], [DCANK17], [SKKK16], [SKK15a], [SKK15b], [RKK15], [RKK14], [RAKK13], [ARKK13], [SRAKK13a], [SRAKK13b],[RAKK12a], [RAKK12b], [ARKK12], [RKK11], [ARKK11], [WKK11], [DRKK10], [WKK08].
Power and Temperature Aware Systems: [CKK16], [PKK13], [KoKr11], [HaKo07], [HKK07], [WGKK06], [GKK05], [UnKo03], [URKK01].
Secure Cryptogrpahic devices: [MDNK17], [BHB14], [MTMK13], [RBIK12], [BBKN12], [BPBKR10], [REBIK08], [ABPK07], [REGB07], [BNK07], [BBKMP03a], [BBKMP03b], [BBKMP03].
Cyber-Physical Systems: [XKK20a], [XKK20b], [KK17], [XKK17], [CKK16], [KK13], [JKKK12], [XKK12].
Fault-Tolerant Systems: [KoKr20], [RKK12], [RKK11], [ZLKK07], [UnKo03], [DuKK01], [HLKK00], [ACDH00], [KKKr98], [YuKo95].
Defect and fault tolerance [LCCTK10], [LCKK09], [ChKo04], [SiKo03], [KoKo00].
Digital computer arithmetic: [WK16], [WiAK14], [Kore02], [WoKo05], [KKO03], [PhGK01], [PhKo99].

Selected List of Recent Publications (copyright note)

Publications on Google Scholar.

[KoKr20] Textbook: *I. Koren and C. M. Krishna, Fault-Tolerant Systems.*, 2nd Edition, **Morgan-Kaufman**, San Francisco, CA, 2020.
[Kore02] Textbook: *I. Koren, Computer Arithmetic Algorithms, Second Edition*, **A. K. Peters**, Natick, MA, 2002.
[XKK21] S. Xu, I. Koren and C.M. Krishna, "Adaptive Workload Adjustment for Cyber-Physical Systems using Deep Reinforcement Learning," (pdf file) *Sustainable Computing: Informatics and Systems*, Vol. 30, 8pp, Ju 2021.
[XKK20a] S. Xu, I. Koren and C.M. Krishna, "Thermal Aware Task Scheduling for Enhanced Cyber-Physical Systems Sustainability," (pdf file) *IEEE Trans. on Sustainable Computing*, Vol.5, pp. 581-593, Oct. 2020.
[XKK20b] S. Xu, I. Koren and C.M. Krishna, "Enhancing Dependability and Energy Efficiency of Cyber-Physical Systems by Dynamic Actuator Derating," (pdf file) *Sustainable Computing: Informatics and Systems*, vol. 28, pp. 1-10, Dec. 2020.
[RKK19] N. Ravi, C. M. Krishna and I. Koren, "Enhancing Vehicular Anonymity in ITS: A New Scheme for Mix Zones and their Placement," (pdf file) *IEEE Trans. on Vehicular Technology*, Vol. 68, no. 11, pp. 10372-10381, Nov. 2019.
[SLKKB18] A. Sartor, A. Lorenzon, S. Kundu, I. Koren and A. Beck, "Adaptive and Polymorphic VLIW Processor to Optimize Fault Tolerance, Energy Consumption, and Performance," (pdf file) *Proc. of the ACM International Conference on Computing Frontiers (CF-2018)*, 8pp., May 2018 (won the Best Paper Award).
[KK17] C. M. Krishna and I. Koren, "Thermal-Aware Management Techniques for Cyber-Physical Systems," (pdf file) *Sustainable Computing: Informatics and Systems*, pp. 39-51, Sept. 2017.
[DCANK17] M. Diener, E. Cruz, M. Alves, P. Navaux and I. Koren, "Affinity-Based Thread and Data Mapping in Shared Memory Systems," (pdf file) *ACM Computing Surveys*, Vol.49, pp. 64.1-64.38, Jan. 2017.
[XKK17] Y. Xu, I. Koren and C.M. Krishna, "AdaFT: A Framework for Adaptive Fault Tolerance for Cyber-Physical Systems," (pdf file) *ACM Transactions on Embedded Computing Systems (TECS)*, pp. 79.1-79.25, April 2017.
[MDNK17] F. Moreira, M. Diener, P. Navaux and I. Koren, "Data Mining the Memory Access Stream to Detect Anomalous Application Behavior," (pdf file) *Proc. of the 2017 ACM Intern. Conf. on Computing Frontiers*, pp. 45-52, May 2017.
[CTTKK16] G. Chapman, R. Thomas, R. Thomas, I. Koren and Z. Koren, "Experimental Study and Analysis of Soft and Permanent Errors in Digital Cameras," (pdf file) *Proc. of the 2016 IEEE Internl. Symp. on Defect and Fault Tolerance in VLSI & Nanotechnology Systems*, 4 pp, Sept. 2016.
[OPKR16] D. Oliveira, L. Pilla, F. Fernandes, C. Lunardi, I. Koren, P. Navaux, L. Carro and P. Rech, "Input Size Effects on the Radiation-Sensitivity of Modern Parallel Processors," (pdf file) *Proc. of the Radiation Effects Data workshop*, 6 pp., July 2016.
[CKK16] M. Chhablani, I. Koren and C. Krishna, "Online Inertia-based Temperature Estimation for Reliability Enhancement" (pdf file) *ASP Journal of Low Power Electronics - JOLPE*, Vol. 12, No. 3, pp. 159-171, Sept. 2016.
[CTTKK16] G. Chapman, R. Thomas, R. Thomas, K. Meneses, T. Yang, I. Koren and Z. Koren, "Increases in Hot Pixel Development Rates for Small Digital Pixel Sizes," (pdf file) *Proc. of the 2016 Symposium on Electronic Imaging*, 5 pp., Feb. 2016.
[MADNK16] F. Moreira, M. Alves, M. Diener, P. Navaux and I. Koren, "A Dynamic Block-Level Execution Profiler," (pdf file) *Parallel Computing*, pp. 15-28, May 2016.
[WK16] S. Wimer and I. Koren, "Energy Efficient Deeply Fused Dot-Product Multiplication Architecture," (pdf file) *Proc. of ASAP'16 - the Internl. Conf. on Application-Specific Systems, Architectures and Processors*, pp. 115-122, July 2016.
[SKKK16] S. Srinivasan, N. Kurella, I. Koren and S. Kundu, "Exploring Heterogeneity within a Core for Improved Power Efficiency," (pdf file) *IEEE Transactions on Parallel and Distributed Systems*, pp. 1057-1068, April 2016.

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/