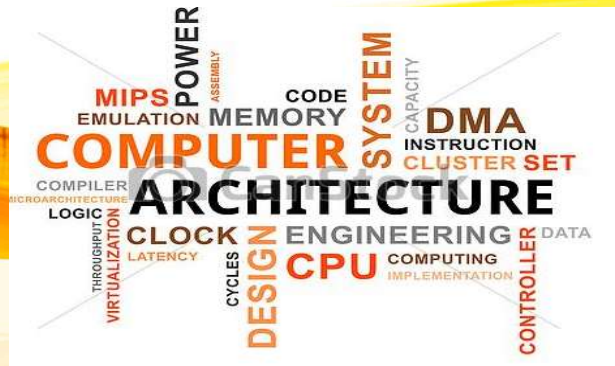
UNIT II
ARITHMETIC OPERATIONS

Addition and subtraction of signed numbers – Design of fast adders – Multiplication of positive numbers - Signed operand multiplication- **fast multiplication** – Integer division – Floating point numbers and operations



Recap the previous Class

Design of Fast Multiplier

a)Bit-Pair Recoding of Booth's Multiplication

–A technique that halves the maximum number of summands; derived directly from the Booth's algorithm.

–If we **group the Booth-coded multiplier digits in pairs**, we observe:

(+1, -1): (+1, -1) * M = 2 * M – M = M

(0, +1): (0, +1) * M = M

–We need a single addition instead of a pair of addition & subtraction.

- Other similar rules can be framed.

Yi+1	Yi	Yi-1	Partial Products
0	0	0	0*Multiplicand
0	0	1	1*Multiplicand
0	1	0	1*Multiplicand
0	1	1	2*Multiplicand
1	0	0	-2*Multiplicand
1	0	1	-1*Multiplicand
1	1	0	-1*Multiplicand
1	1	1	-0*Multiplicand

Design of Fast Multiplier

Original Booth-coded Pair	Equivalent Recoded Pair
(+1, 0)	(0, +2)
(-1, +1)	(0, -1)
(0, 0)	(0, 0)
(0, 1)	(0, 1)
(+1, 1)	--
(+1, -1)	(0, +1)
(-1, 0)	(0, -2)

- Every equivalent recoded pair has at least one 0.
- Worst-case number of additions or subtractions is 50% of the number of multiplier bits.
- Reduces the worst-case time required for multiplication.

Example:

Original:	Multiplier --	1	0	1	0	1	0
Booth:	Multiplier --	-1	+1	-1	+1	-1	0
Recoded:	Multiplier --	0	-1	0	-1	0	-2

```
          0  0  1  1  0  1
          . -1  . -1  . -2
                 -------
-----------------
1 1 1 1 1 1 1 0      -
                  0 1 1 0
  1 1 1 1 1 1 0 0 1 1
  1 1 1 1 0 0 1 1
_____________________________
  1 1 0 1 1 1 1 0 0 0 1 0
```

M = 0 0 1 1 0 1 (+13)

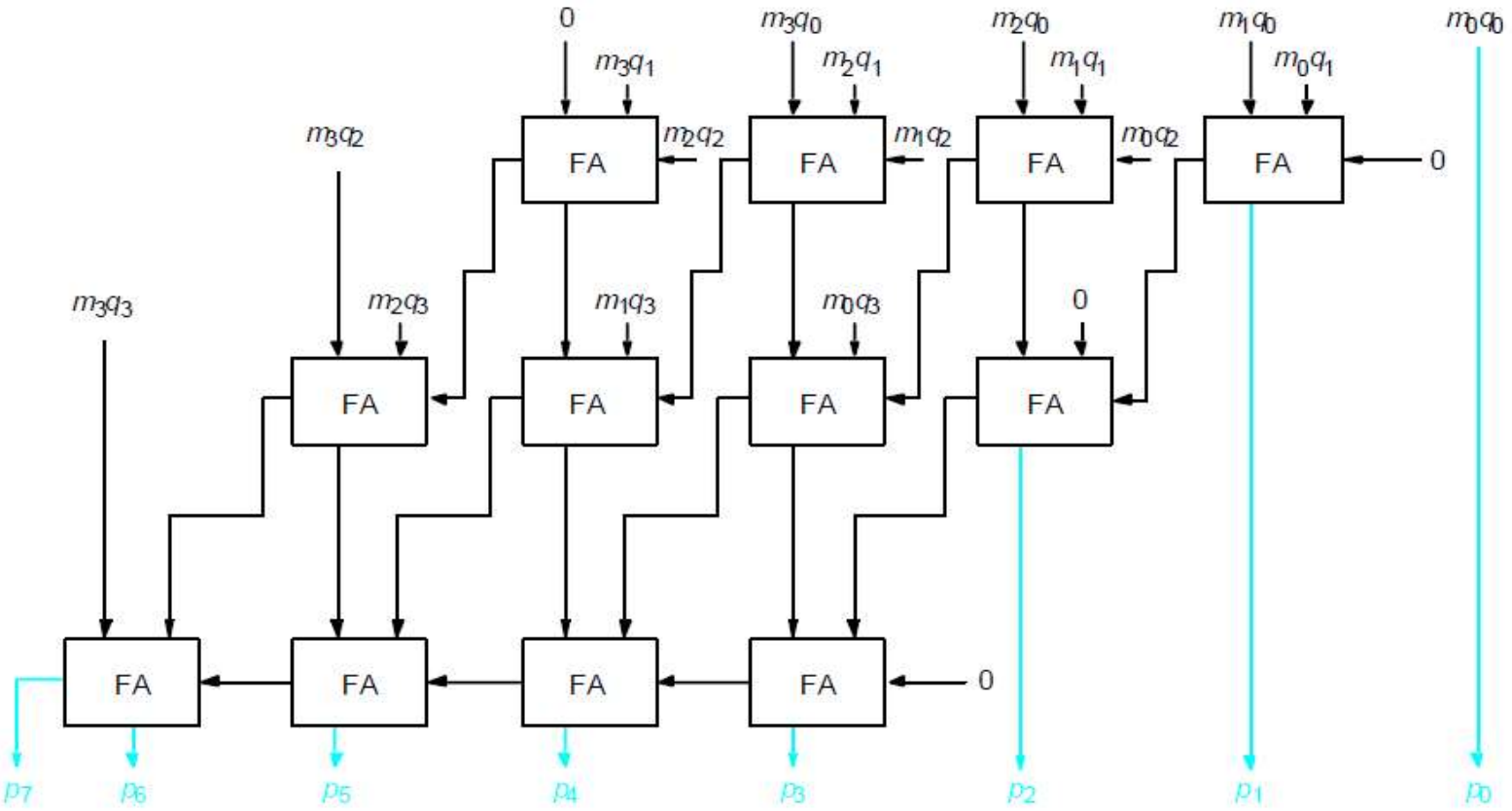
-1 * M = 1 1 0 0 1 1

-2 * M = 1 0 0 1 1 0

b) Carry Save Multiplier

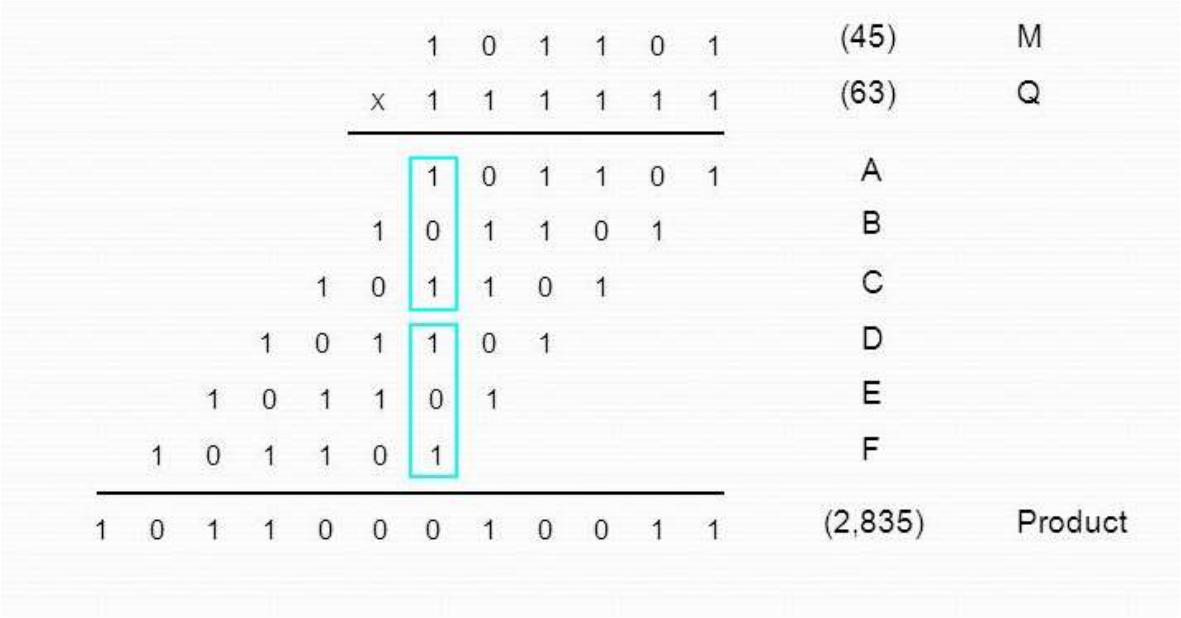
- We have seen earlier how carry save adders (CSA) can be used to add several numbers with carry propagation only in the last stage.
- The **partial products** can be generated in **parallel using n^2 AND gates**.
- The n partial products can then be added using a **CSA tree.**
- Instead of letting the carries ripple through during addition, we ***save* them and feed it to the next row**, at the correct weight positions.

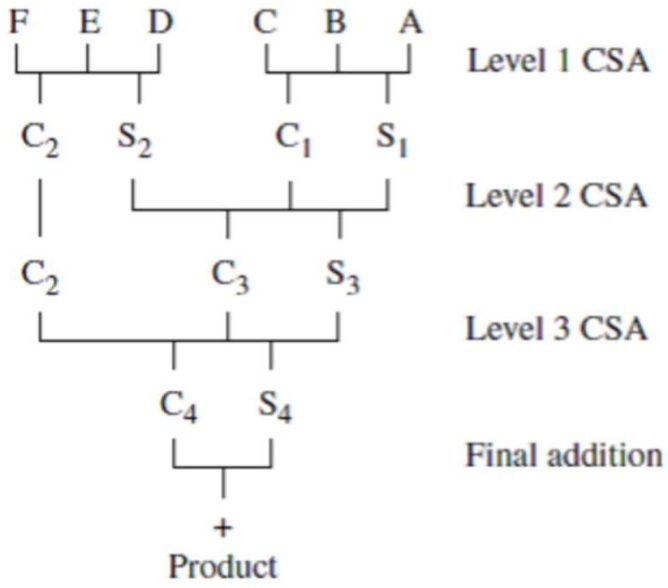
4 x 4 Carry Save Multiplier

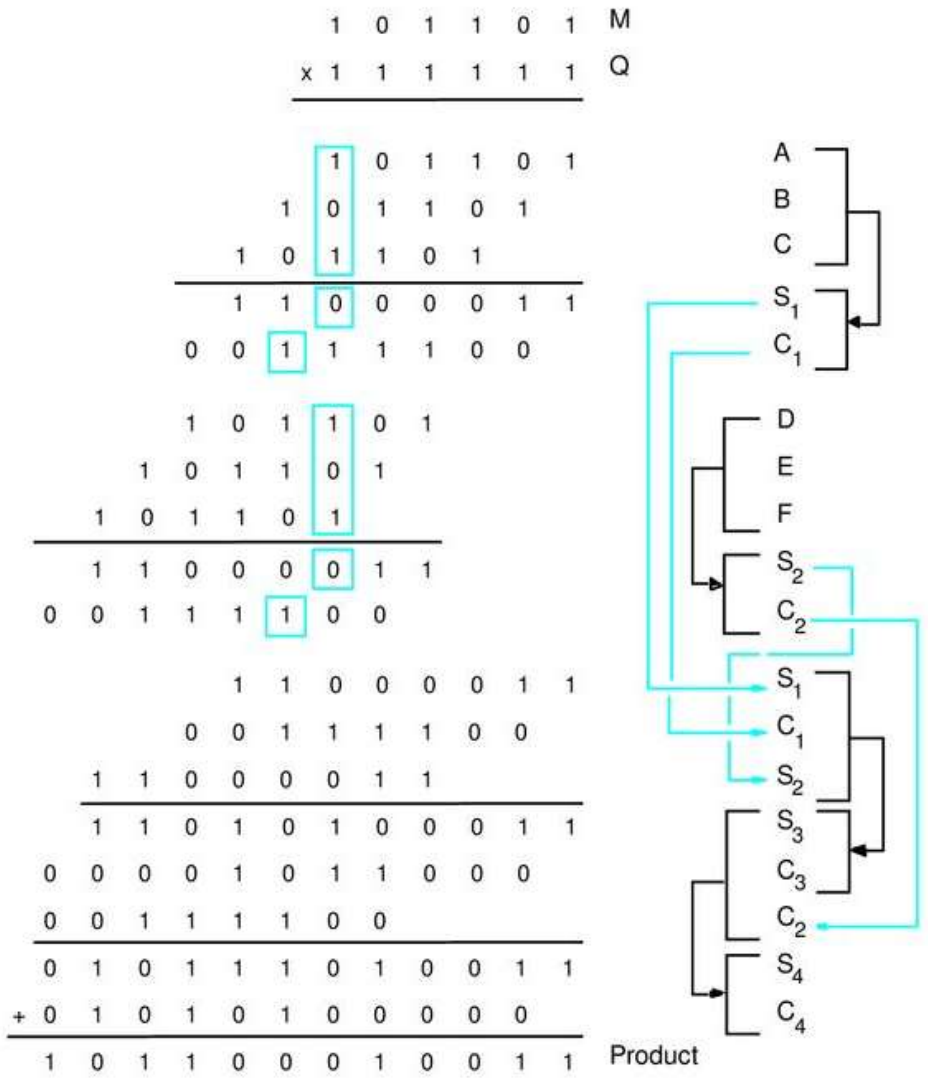


Example

- Consider the number 45 x 63. Perform Carry save addition




F E D C B A
Level 1 CSA
C2 S2 C1 S1
Level 2 CSA
C2 C3 S3
Level 3 CSA
C4 S4
Final addition
+
Product

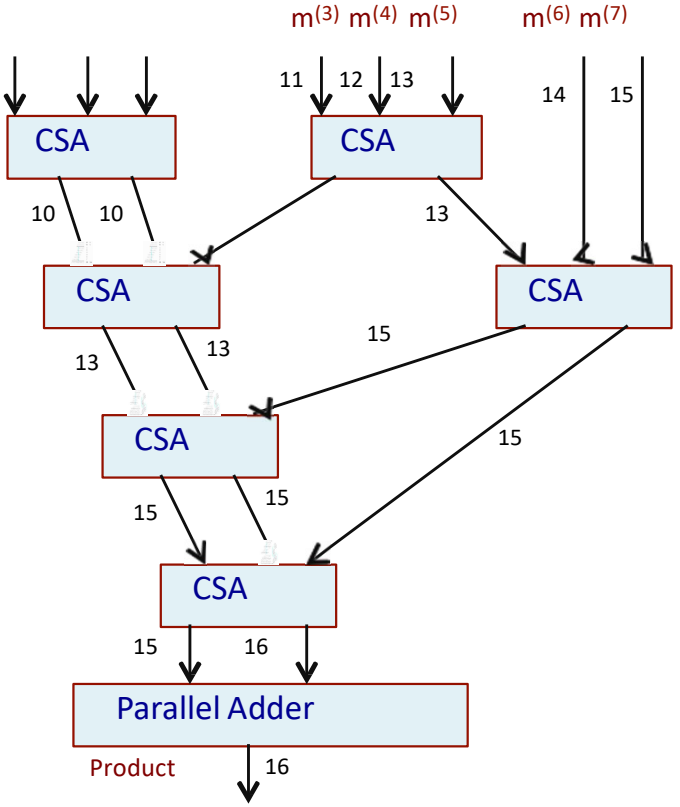

M
Q
Product

A
B
C
S1
C1
D
E
F
S2
C2
S1
C1
S2
S3
C3
C2
S4
C4

Wallace Tree Multiplier

- A Wallace tree is a circuit that reduces the problem of summing n n-bit numbers to the problem of summing two Θ(n)-bit numbers.
- It uses n/3 (floor of) carry-save adders in parallel to convert the sum of n numbers to the sum of 2n/3 (ceiling of) numbers.
- It then recursively constructs a Wallace tree on the 2n/3 (ceiling of) resulting numbers.
- The set of numbers is progressively reduced until there are only two numbers left.
- By performing many carry-save additions in parallel, Wallace trees allow two n-bit numbers to be multiplied in $\Theta(\log_2 n)$ time using a circuit of size $\Theta(n^2)$.

- The figure shows a Wallace tree that adds 8 partial products $m^{(0)}$, $m^{(1)}$, …, $m^{(7)}$.
- The partial product $m^{(i)}$ consists of (n+i) bits.
- Each line represents an entire number – the label of an edge indicates the number of bits.
- The carry-lookahead adder at the bottom adds a (2n-1)-bit number to a 2n-bit number to give the 2n-bit product.



TEXT BOOK

Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, McGraw-Hill, 6th Edition 2012.

REFERENCES

1. David A. Patterson and John L. Hennessey, “Computer organization and design”, MorganKauffman ,Elsevier, 5th edition, 2014.
2. William Stallings, “Computer Organization and Architecture designing for Performance”, Pearson Education 8th Edition, 2010
3. John P.Hayes, “Computer Architecture and Organization”, McGraw Hill, 3rd Edition, 2002
4. M. Morris R. Mano “Computer System Architecture” 3rd Edition 2007
5. David A. Patterson “Computer Architecture: A Quantitative Approach”, Morgan Kaufmann; 5th edition 2011

THANK YOU