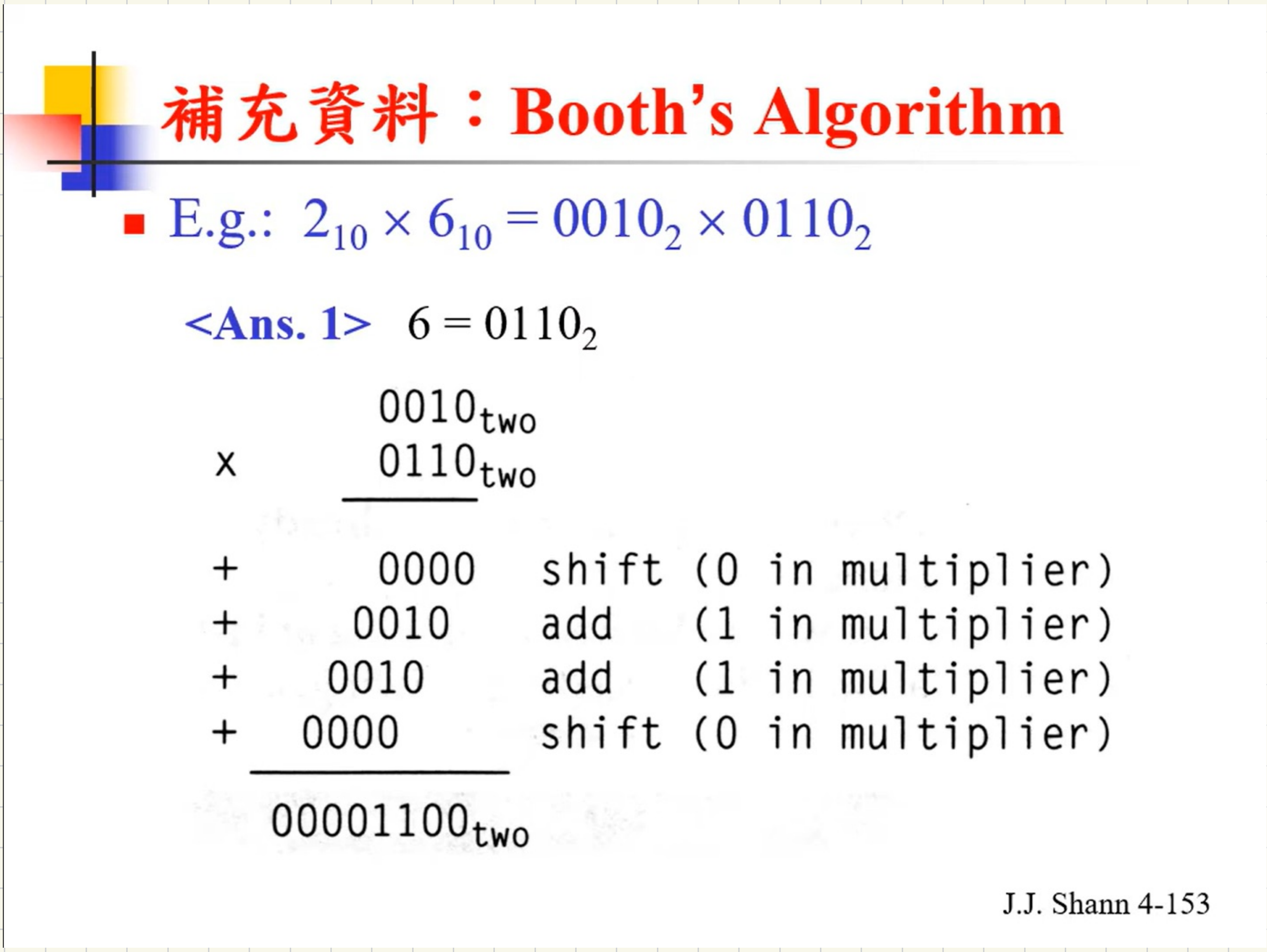
4.10.2 Booth's Multiplier



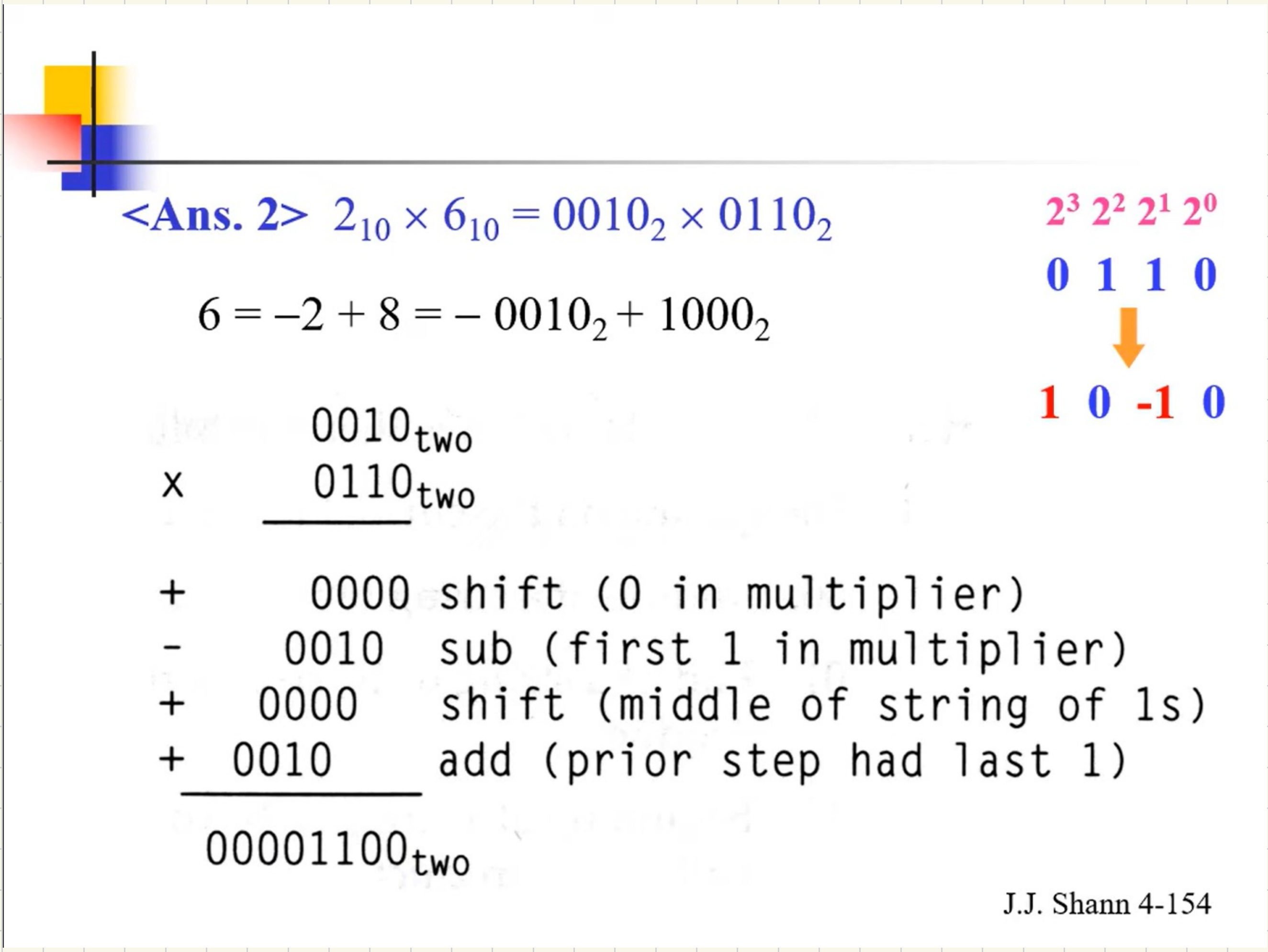
補充資料：Booth's Algorithm

- E.g.: $2_{10} \times 6_{10} = 0010_2 \times 0110_2$

<Ans. 1> $6 = 0110_2$

```
             0010two
  x          0110two
          ----------
  +          0000    shift (0 in multiplier)
  +         0010     add   (1 in multiplier)
  +        0010      add   (1 in multiplier)
  +       0000       shift (0 in multiplier)
      --------------
        00001100two
```

J.J. Shann 4-153



<Ans. 2> $2_{10} \times 6_{10} = 0010_2 \times 0110_2$

$6 = -2 + 8 = -0010_2 + 1000_2$

$2^3\ 2^2\ 2^1\ 2^0$

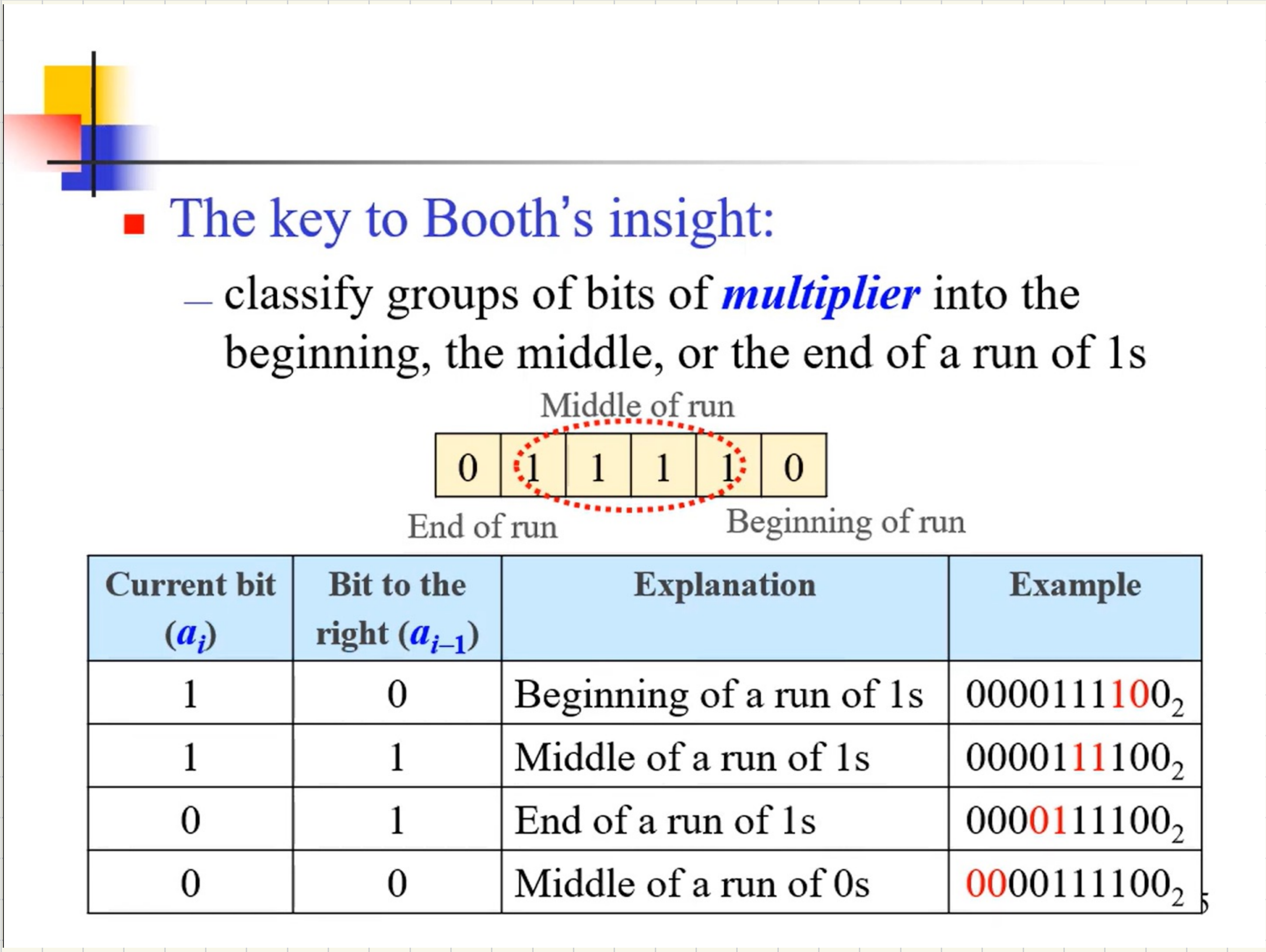
0 1 1 0

↓

1 0 -1 0

```
           0010two
  x        0110two
        ----------
  +        0000  shift (0 in multiplier)
  -       0010   sub (first 1 in multiplier)
  +      0000    shift (middle of string of 1s)
  +     0010     add (prior step had last 1)
      ------------
       00001100two
```

J.J. Shann 4-154

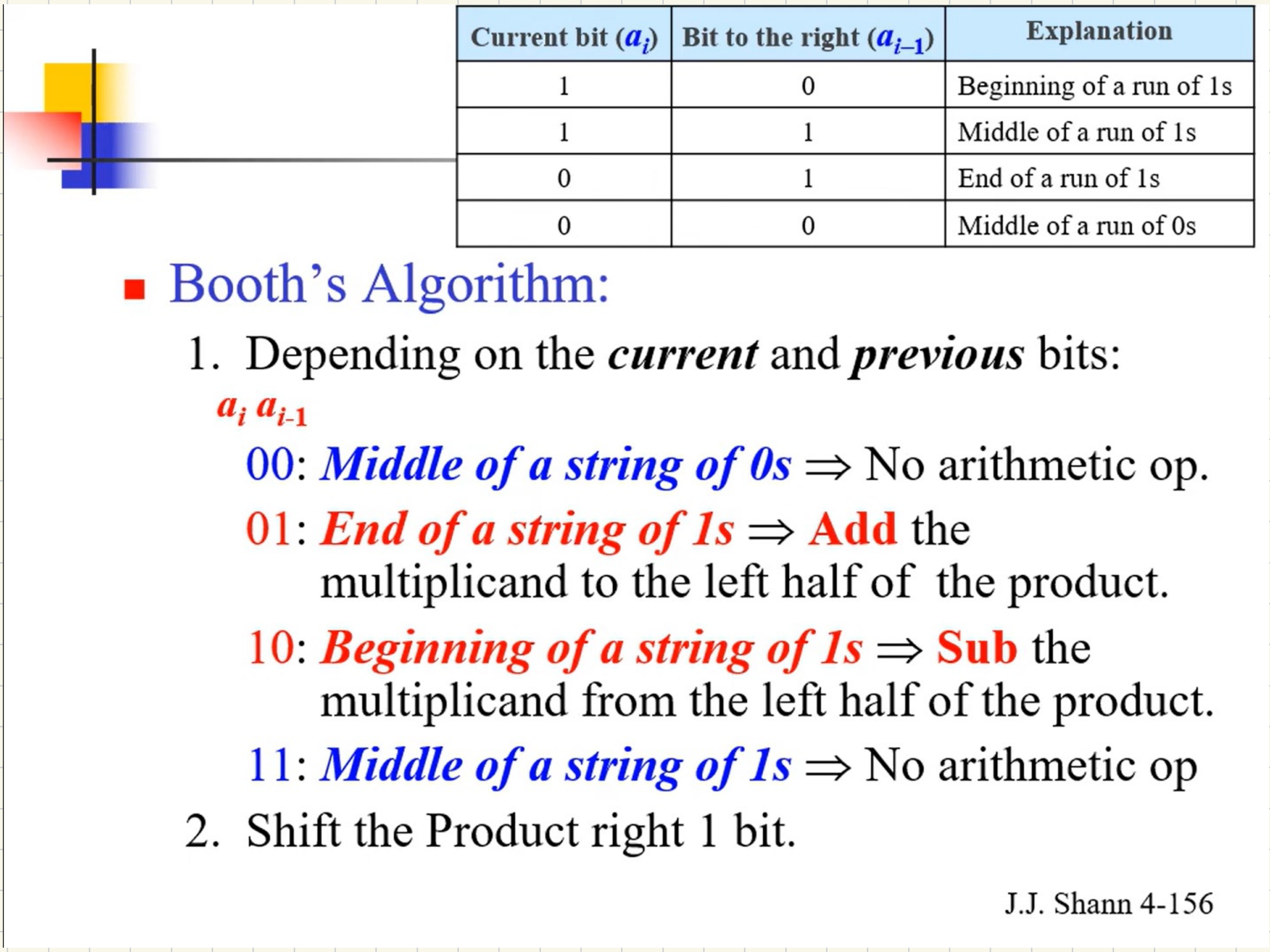


The key to Booth's insight:

– classify groups of bits of ***multiplier*** into the beginning, the middle, or the end of a run of 1s

Middle of run
0
1
1
1
1
0
End of run
Beginning of run

Current bit (a_i)	**Bit to the right (a_{i-1})**	**Explanation**	**Example**
1	0	Beginning of a run of 1s	0000111100_2
1	1	Middle of a run of 1s	0000111100_2
0	1	End of a run of 1s	0000111100_2
0	0	Middle of a run of 0s	0000111100_2



Current bit (a_i)	**Bit to the right (a_{i-1})**	**Explanation**
1	0	Beginning of a run of 1s
1	1	Middle of a run of 1s
0	1	End of a run of 1s
0	0	Middle of a run of 0s

Booth's Algorithm:

1. Depending on the ***current*** and ***previous*** bits:

   $a_i\ a_{i\text{-}1}$

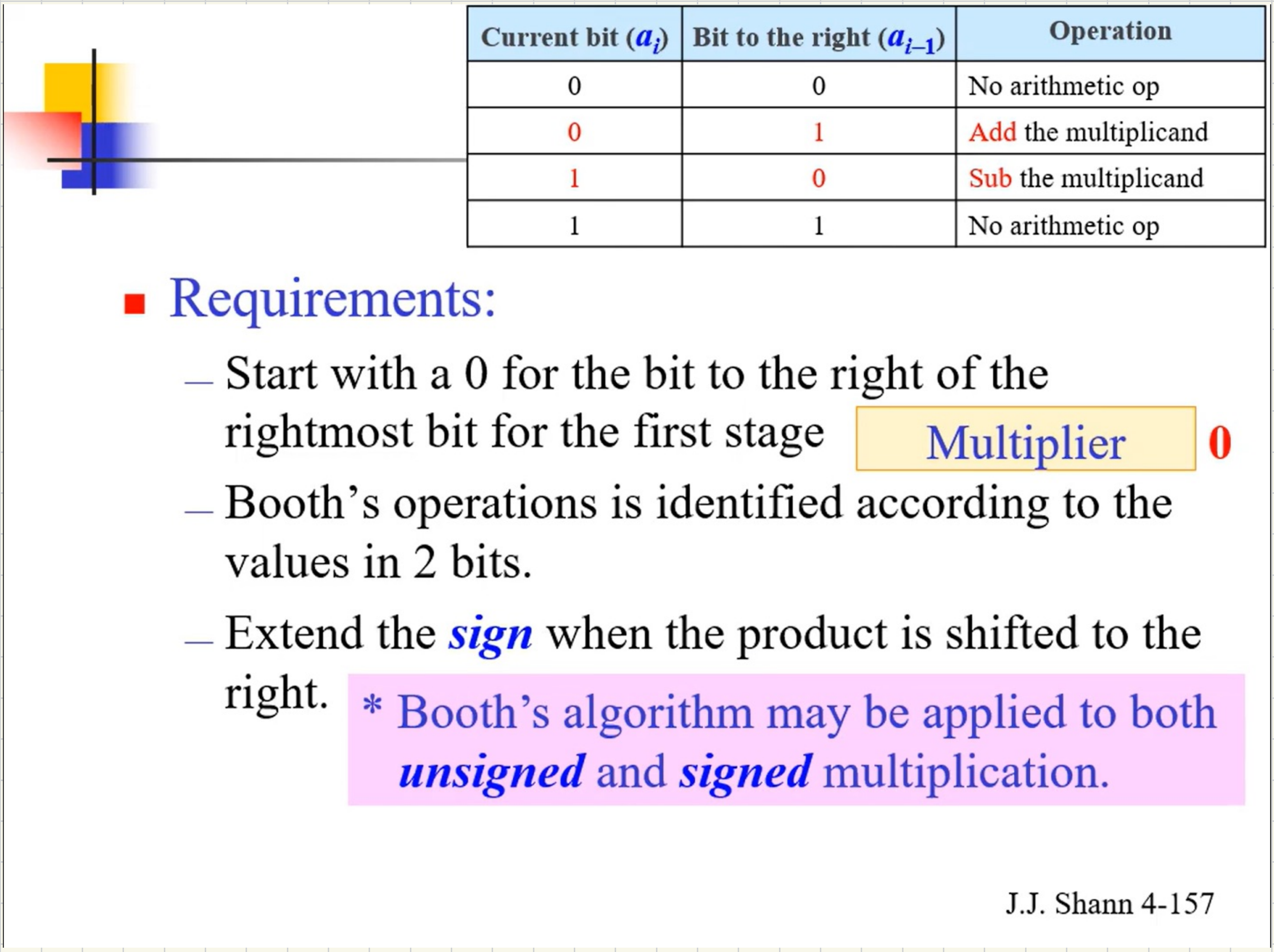
   00: ***Middle of a string of 0s*** ⇒ No arithmetic op.

   01: ***End of a string of 1s*** ⇒ **Add** the multiplicand to the left half of the product.

   10: ***Beginning of a string of 1s*** ⇒ **Sub** the multiplicand from the left half of the product.

   11: ***Middle of a string of 1s*** ⇒ No arithmetic op

2. Shift the Product right 1 bit.

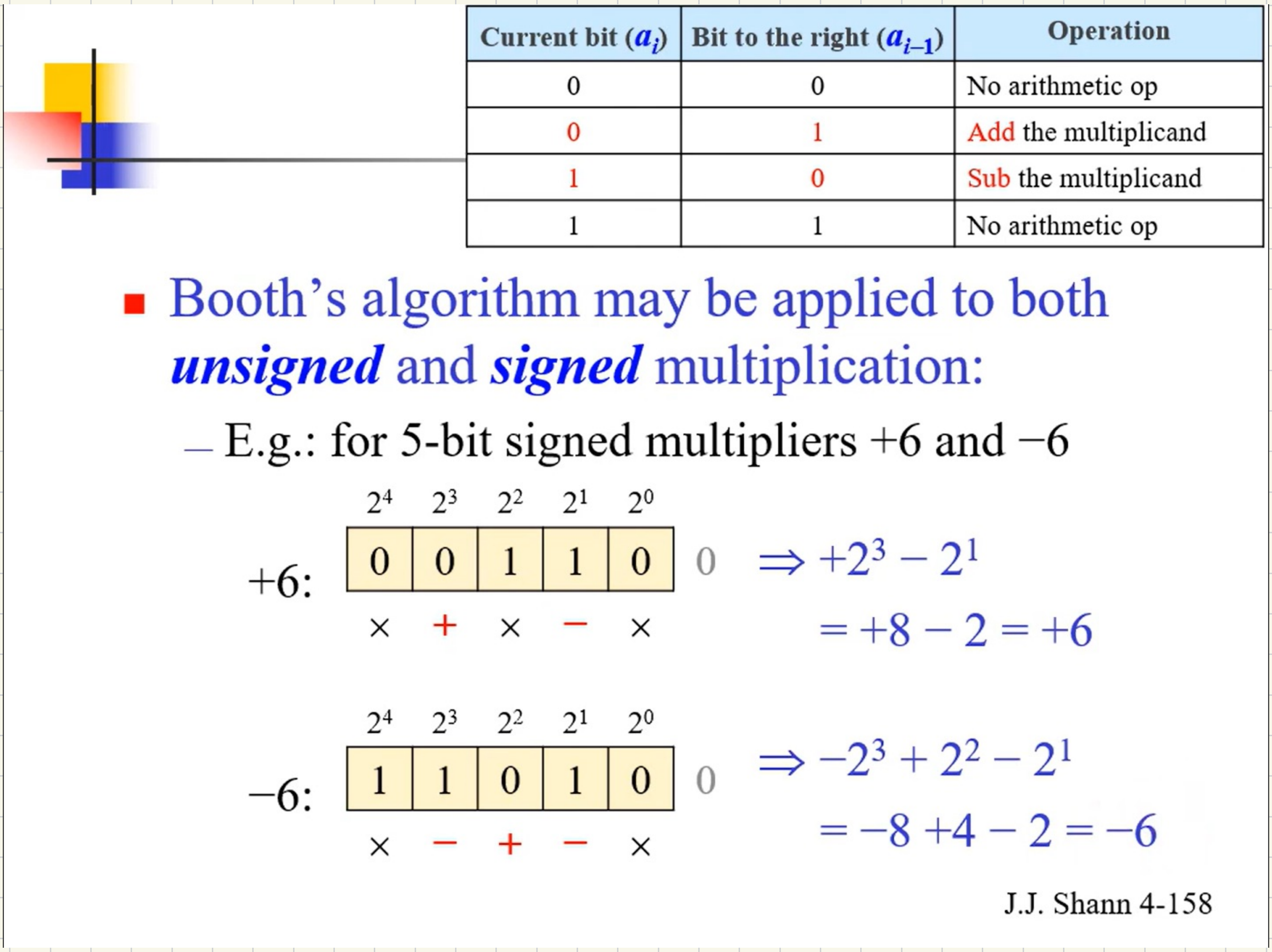


Current bit (a_i)	Bit to the right (a_{i-1})	Operation
0	0	No arithmetic op
0	1	Add the multiplicand
1	0	Sub the multiplicand
1	1	No arithmetic op

- Requirements:
  - Start with a 0 for the bit to the right of the rightmost bit for the first stage
  - Booth's operations is identified according to the values in 2 bits.
  - Extend the ***sign*** when the product is shifted to the right.

Multiplier	0

* Booth's algorithm may be applied to both ***unsigned*** and ***signed*** multiplication.



Current bit (a_i)	Bit to the right (a_{i-1})	Operation
0	0	No arithmetic op
0	1	Add the multiplicand
1	0	Sub the multiplicand
1	1	No arithmetic op

- Booth's algorithm may be applied to both ***unsigned*** and ***signed*** multiplication:
  - E.g.: for 5-bit signed multipliers +6 and −6

	2^4	2^3	2^2	2^1	2^0		
+6:	0	0	1	1	0	0	$\Rightarrow +2^3 - 2^1$
	×	+	×	−	×		$= +8 - 2 = +6$

	2^4	2^3	2^2	2^1	2^0		
−6:	1	1	0	1	0	0	$\Rightarrow -2^3 + 2^2 - 2^1$
	×	−	+	−	×		$= -8 + 4 - 2 = -6$