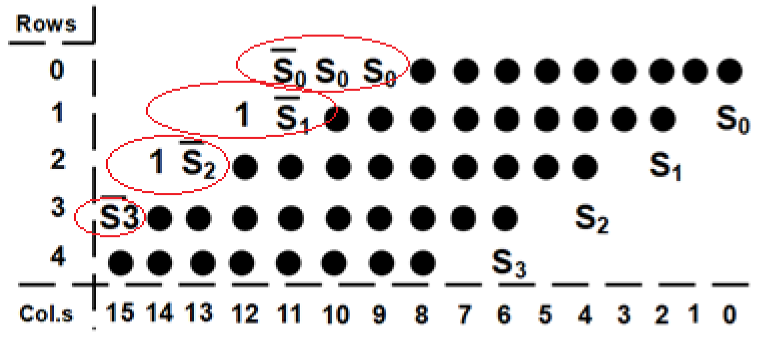
Booth-Multiplier Implementation

Unsigned 8×8 Radix-4 Booth Multiplier dot_diagram (based on the dor-diagram on page 146 of “CSL-TR-94-617.appendix.pdf”)

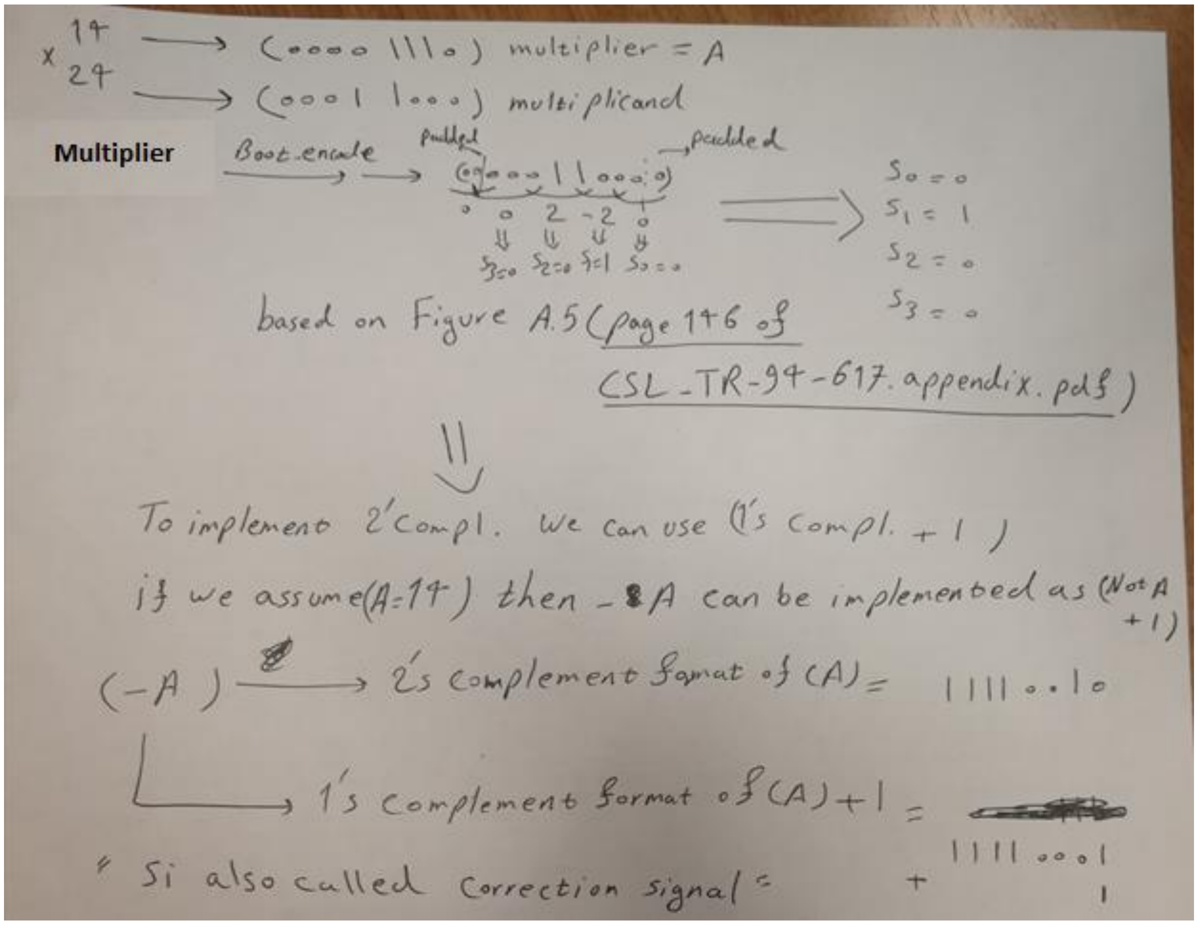
S_i = sign of partial product of ith row



Rows
0 $\overline{S_0}$ S_0 S_0 ● ● ● ● ● ● ● ● ●
1 1 $\overline{S_1}$ ● ● ● ● ● ● ● ● ● S_0
2 1 $\overline{S_2}$ ● ● ● ● ● ● ● ● ● S_1
3 $\overline{S3}$ ● ● ● ● ● ● ● ● ● S_2
4 ● ● ● ● ● ● ● ● S_3
Col.s 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Figure 1- Internal architecture of an unsigned radix-4 Booth multiplier

The following pictures show an example which is based on the dot-diagram shown in Figure 1.



14 → (0000 1110) multiplier = A
x 24 → (0001 1000) multiplicand

Multiplier Boot encode → (0 0000 1100 0 0) padded, padded

0 0 2 -2 0

$S_3 = 0$ $S_2 = 0$ $S_1 = 1$ $S_0 = 0$

⟹ $S_0 = 0$, $S_1 = 1$, $S_2 = 0$, $S_3 = 0$

based on Figure A.5 (page 146 of CSL-TR-94-617.appendix.pdf)

⇓

To implement 2'compl. we can use (1's compl. + 1)

if we assume (A=14) then -A can be implemented as (Not A + 1)

(-A) → 2's complement format of (A) = 1111 0010

└→ 1's complement format of (A) + 1 = 1111 0001 + 1

* Si also called correction signal

$S_0 = 0$, $S_1 = 1$, $S_2 = 0$, $S_3 = 0$

```
14 = (A)              S̄0 S0 S0 0  0 000 0 000          (0×A)
×24                                                      ⊕
                  1 S̄1 1 1 1  1 000  1 1 ← S0        (−2×A) = Not(2A)+1
         1 S̄2 0  0  0  0 1  1 1 0 0        S1 ←          ⊕
S̄3  0  0  0  0  0  0  0  0  0 ← S2                     (2×A)
                                                         ⊕
0  0  0  0  0  0  0  0 ← S3                             (0×A)
                                                         ⊕
                                                        (0×A)
```

⇓ applying the values of $S_0 \dots S_3$

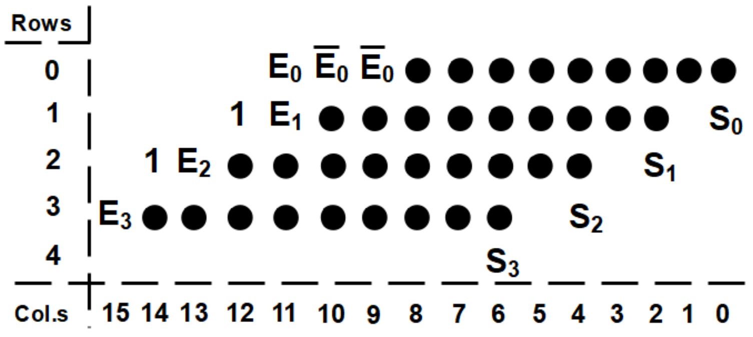
```
                 1 0 0 0 0 0 0 0 0 0 0 0
              1 0 1 1 1 1 0 0 0 1 1   0
 +       1 1 0 0 0 0 1 1 1 0 0   1
      1 0 0 0 0 0 0 0 0 0   0
      0 0 0 0 0 0 0 0   0
 ---------------------------------------
      0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0
```

$= 336$

Signed 8×8 Radix-4 Booth Multiplier dot_diagram (based on the dor-diagram on page 146 of "CSL-TR-94-617.appendix.pdf")

S_i = sign of partial product of ith row

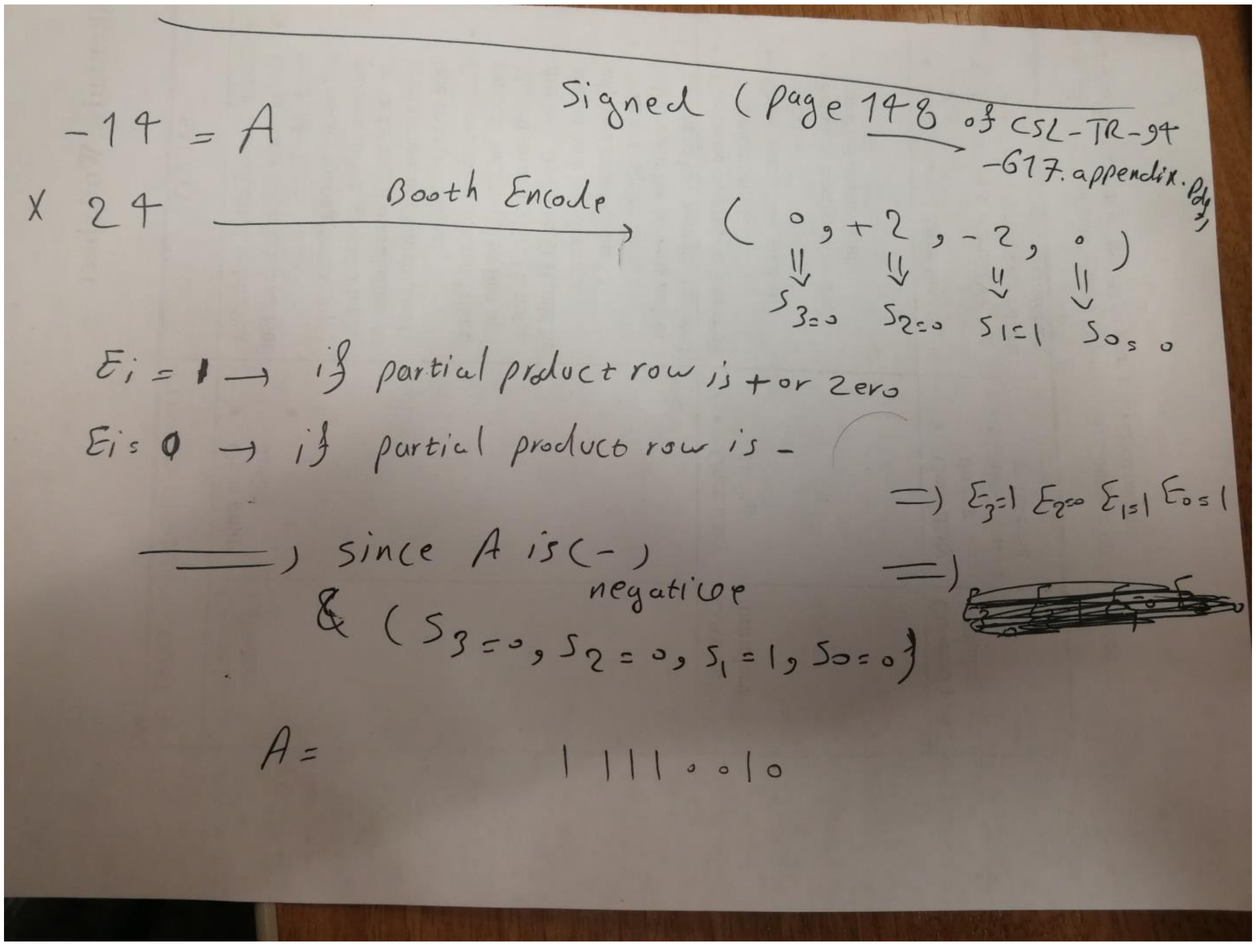
E_i = S_i XNOR multiplicand sign



Rows
0
1
2
3
4
E_0 $\overline{E_0}$ $\overline{E_0}$
1 E_1
1 E_2
E_3
S_0
S_1
S_2
S_3
Col.s 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Figure 2- Internal architecture of a signed radix-4 Booth multiplier

The following pictures show an example which is based on the dot-diagram shown in Figure 2.



Signed (Page 148 of CSL-TR-94-617.appendix.pdf)

$-14 = A$

$\times\ 24 \xrightarrow{\text{Booth Encode}} (0, +2, -2, 0)$

$\Downarrow$ $S_3=0$ $\Downarrow$ $S_2=0$ $\Downarrow$ $S_1=1$ $\Downarrow$ $S_0=0$

$E_i = 1 \rightarrow$ if partial product row is + or zero

$E_i = 0 \rightarrow$ if partial product row is −

$\Rightarrow E_3=1\ E_2=0\ E_1=1\ E_0=1$

$\Rightarrow$ since A is (−) negative

& ($S_3=0$, $S_2=0$, $S_1=1$, $S_0=0$)

$\Rightarrow$ [illegible]

$A =$ 1111 0010

$E_0\ \bar{E}_0\ \bar{E}_0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$

$1\ E_1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ S_0$ $(0 \times A)$

$1\ E_2\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ S_1$ $(-2 \times A)$

$E_3\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ S_2$ $(2 \times A)$

S_3 $(0 \times A)$

$\Downarrow$ applying E_i, S_i

$1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$

$+\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1$

$1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1$

$1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$

0

$$\underbrace{1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0}$$

$= -336$

Hints:

1. **The partial product row bits shown in Figures 1 and 2 (circles) are 9 bits in each row since to shift the multiplicand when the booth encoded element is +2 or -2. In case the Booth encoded element is 0,+1,-1 this bit (most significant bit) will be filled with sign bit.**
2. **As mentioned in the "Dot_Digram.pdf", in the given Booth multiplier Verilog file, S_i signals are also called neg_i or cor_i.**
3. **In case, you want to change the given unsigned multiplier, you need to change the S_i signals and "1s" (those that are distinguished with red circle in Figure 1) to E_i and "1s" signals (that already explained both in this document and CSL-TR-94-617.appendix.pdf page 148). Also you have to remove the extra partial product row (partial product row 4 except S_3). However, if you can's change it, you still can use the given code. In more detail, since the inputs are in signed format, you have to convert them to unsigned format, then change the multiplication result accordingly if needed (for example, if one of the operand is + and the other one is -).**
4. **Note that, the unsigned Booth format (figure 1) has and extra partial product row (row 4 except S_3).**
5. **Figure 3 shows the dot_diagram the of unsigned Booth multiplier when approximation in topic 3 is applied.**

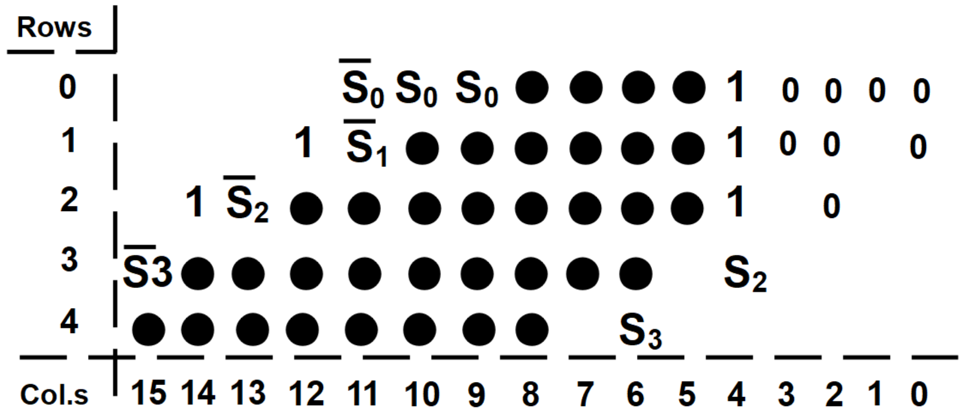

Rows
0
1
2
3
4
$\overline{S}_0$ S_0 S_0 ● ● ● ● 1 0 0 0 0
1 $\overline{S}_1$ ● ● ● ● ● ● 1 0 0 0
1 $\overline{S}_2$ ● ● ● ● ● ● ● ● 1 0
$\overline{S}3$ ● ● ● ● ● ● ● ● ● S_2
● ● ● ● ● ● ● ● S_3
Col.s
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Figure 3- Internal architecture of an unsigned radix-4 Booth multiplier when the 5 least-significant column are replaced with "1"

6. **To apply approximation in topic 4, you have to implement the approximate half adders and full adders, then replace those exact half adders and full adders in the 4 least-significant bits with approximate ones.**
7. **To apply approximation in topic 5, you have to implement the approximate partial product generator, then replace those exact partial product generators in the 5 least-significant columns.**
8. **In the given unsigned Booth multiplier Verilog file, the partial products are implemented using forgenerate. To implement the approximation in topic 3, and 5, you may change this part of the code. And to implement the approximation in topic 4, you simply need to change the half adders and full adders of the 5 least-significant columns in the accumulation stages.**

You can email m.asadi@usask.ca if you have any questions.