# Product Select Multiplier

Craig M. Conway
National Instruments
6504 Bridge Point Parkway
Austin, Texas 78730

Earl E. Swartzlander, Jr.
The University of Texas at Austin
Austin, Texas

## Abstract

*The Booth multiplier represents an efficient and simple way to multiply signed binary numbers. This paper describes an improvement to the standard implementation of a Booth multiplier at any radix. More specifically, it compares VHDL implementations of conventional Radix 2, 4, 8, 16, and 32 Booth multipliers against multipliers with the product select circuitry added. VHDL models of the multipliers are analyzed and conclusions are drawn. Estimates of area gain and speedup indicate that this multiplier scheme has significant advantages at lower radices and, at higher radices, still maintains a relative advantage over the Radix 2 implementation.*

## 1 Introduction

Booth [1] describes a simple algorithm whereby two numbers may be multiplied without requiring special circuitry to provide a correction factor if either or both of those numbers are negative. Booth's algorithm travels along the bits of the multiplier (M), using each bit and the next least significant bit to determine what multiple of the multiplicand (R) should be used to create a new partial product. Control logic determines from the pair of bits of M ($M_i$ and $M_{i+1}$) whether to add R to the partial product, subtract R from the partial product, or perform no operation at all. The appropriate value is routed through a multiplexer to the adder, and that value is added to the partial product from the previous operation. After shifting the new partial product one place to the right, the process repeats with the next most significant bit ($M_{i-1}$ and $M_i$) so that the next pair can be used to select the proper multiple of R. By shifting one bit at a time, this Radix 2 algorithm requires n clock cycles for an n bit multiplier.

In an effort to improve multiplier performance, MacSorley [2] described the modified Booth algorithm which includes uniform shifts of two and three bits, reducing the required numbers of clock cycles by two or three times, respectively. The modified Booth algorithm provides a table of values of R multiples to be added to the partial product according to the appropriate number bits of the multiplier and its next least significant bit. Each higher radix requires more multiples of R to be provided for addition to the partial product.

As Sam and Gupta show in [3], the modified Booth algorithm can be expanded to create multipliers of Radix 16, 32, 64, and higher. As the radix of the multiplier increases, so does the number of bits of the product that are created each clock cycle. The area of the multiplier also increases, however, because many of the multiples of R themselves require adders to calculate. For example, a Radix 8 multiplier requires the value of 3R, incurring one adder delay in its calculation. Higher radix multipliers, such as Radix 32, require multiples of R that cannot be calculated with a single adder delay (11R requires two adder delays). Thus, while the difference in the amount of circuitry between a Radix-2 and Radix-4 multiplier is relatively small, above Radix-4, each doubling of the radix more than doubles the number of additional adders required to calculate the multiples of R.

The decrease in the number of clock cycles, then, can be partially offset by a slower clock time which may be required. When R is first applied to the circuit, sufficient setup time must be given to allow each multiple of R to be calculated before the multiplexer selects which multiple is routed to the adder to be added to the partial product. Another option is to give the multiplier additional clock cycles to allow the R multiples to be calculated; however, this defeats the initial purpose of using a higher radix multiplier, that being the reduction of the number of clock cycles required to form the product.

## 2 Product Select Multiplier

In order to reduce the overall delay and therefore reduce the cycle time of a Booth multiplier, certain operations that contribute to the delay can be performed in parallel. The multiplier can take advantage of the fact that the R input is an unchanging value throughout the multiplication. The multiplier calculates ±R, ±2R, ±3R, etc., at the beginning of the multiplier cycle and uses these values throughout, each value being selected as needed by the multiplexer control circuitry.

This idea can be extended in a fashion analogous to the operation of the carry select adder discussed by Bedrij [4]. A carry select adder calculates two sum bits for each input bit position. The first assumes the carry



1058-6393/95 $4.00 © 1995 IEEE

input is zero while the second assumes the carry input is a one. The actual carry input signal then controls a multiplexer which selects the appropriate sum.

By the same token, the Booth multiplier can begin calculating each possible new partial product before the control logic has determined exactly which multiple of R needs to be added to the old partial product. As soon as a partial product is formed and latched into its register, separate adders can begin calculating P times R, P times -R, P times 2R, P times -2R, etc. Each of these potential partial products waits at a multiplexer to be selected by the control logic, removing the delay of the control logic from the calculations of overall performance.

To illustrate, a traditional Radix 4 multiplier based on one found in Swartzlander [5] will be discussed along with a modified version using the product select technique. The implementation of Booth multipliers of other Radices will then be discussed. Figure 1 provides a block diagram of a conventional Radix-4 Booth multiplier.

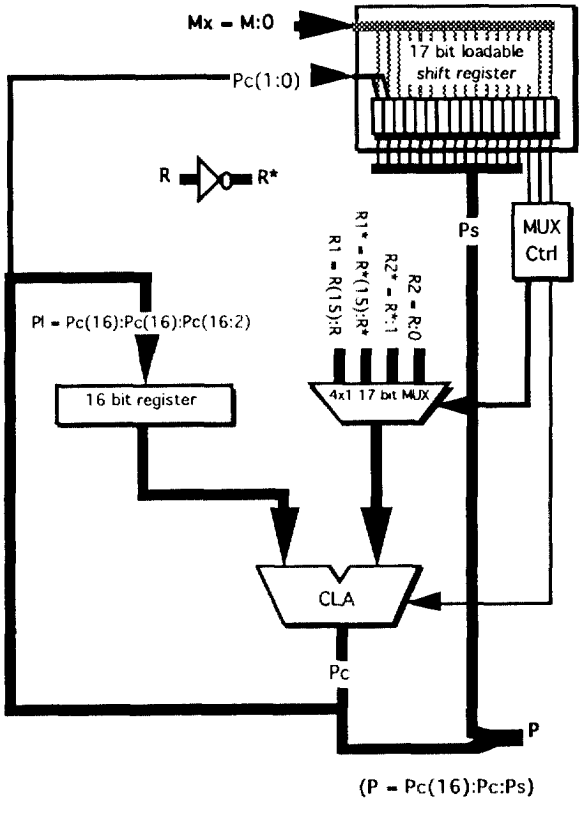


Figure 1: Conventional Radix 4 Booth Multiplier

In Figure 1, M, the multiplier, is concatenated with 0 and latched into the shift register. At the same time, R appears. 2R is calculated instantly while -R and -2R are calculated after one gate delay. The multiplexer control circuitry determines from the lowest two bits of the shift register which multiple of R will enter the adder.

In addition to selecting the multiple of R, the control circuitry also provides a signal to the adder's carry input. When the control logic selects -R or -2R, the one's complement of R or 2R appears at the output of the multiplexer. This signal drives the carry in to the carry lookahead adder (CLA) so that full two's complement addition is performed. Once the CLA adds the old partial product to the appropriate multiple of R, the lower two bits of the new partial product are shifted into the shift register. The upper bits of the new partial product are sign extended and latched into the P register so that the next partial product can be calculated.

A counter (not shown in Figure 1) keeps track of the number of cycles that have taken place and issues a done signal that tells external circuitry that valid data appears at the output.

The product select multiplier approach operates similarly. A block diagram of this multiplier is shown in Figure 2.

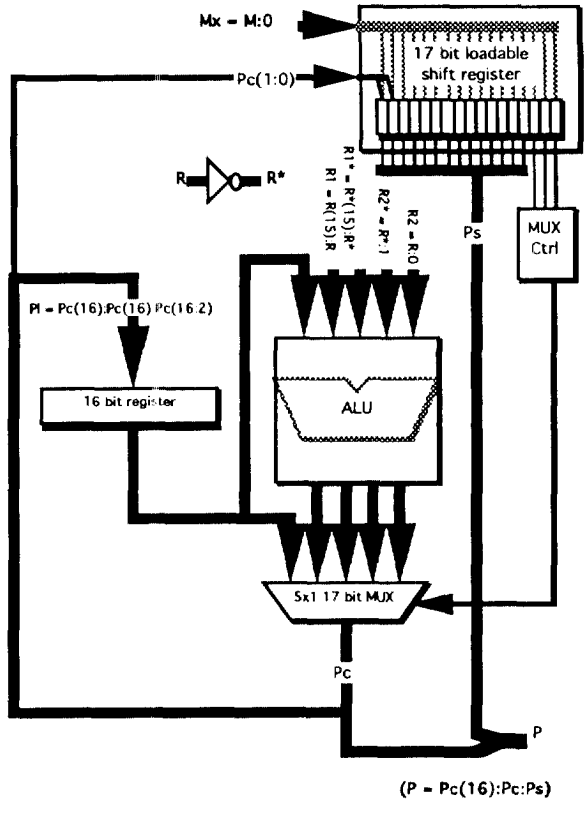


Figure 2: Radix 4 Product Select Booth Multiplier

The multiplier, M, enters the shift register as before and the values of R are calculated; however, the ALU block begins to immediately calculate the sum of the partial product and each multiple of R, creating a total of five potential partial products. Once the control circuitry determines which of the values to select, the multiplexer routes the appropriate partial product to its outputs. The control circuitry itself is similar to that of

the original multiplier except that there is no need to drive the carry input to the adder. Instead, the control circuitry of the product select multiplier requires a signal that causes the multiplexer to select the partial product data without any multiple of R added. The carry signal is not needed because each adder within the ALU has its carry input tied high or low according to whether it has R, 2R, -R, or -2R as one of its inputs (similar to the carry select adder). The multiplexer now requires five input values rather than four, adding an additional gate delay.

In general, an extra gate delay is required on any Radix X multiplier where X is a power of 4. For example, the Radix 16 multiplier has 16 possible R multiples, however, the product select Radix 16 multiplier has 17 potential partial products, incurring an additional OR gate to accommodate the 17th product.

Once the partial product has been calculated, the multiplier operates exactly as the original Radix 4 multiplier in Figure 1. The lower two bits of the partial product are routed to the shift register while the upper bits are sign extended and latched. The same counter can be used to time the cycle and issue a done signal.

## 3 Results

In order to test performance of these multipliers, Viewlogic VHDL was used to implement 16 bit Radix 2, Radix 4, Radix 8, Radix 16, and Radix 32 Booth multipliers. Gate delays and flip-flop clock to output times were assumed to be one nanosecond. Programs were then written in "C" to generate test vectors that were run on Viewlogic simulation tools.

Each of the multipliers has four inputs, a clock, a LOAD signal, M, and R. The output is the product, P. The multipliers expect the LOAD signal to assert to indicate that valid data is on both the M and R buses and that the M value should be latched into the shift register. The multipliers assume that R does not change throughout the cycle and therefore do not latch R. Once the internal counter determines that the product is valid, additional circuitry latches the product. That product is then verified by the simulator. The maximum clock rate for each multiplier was determined by altering the cycle time at which the test vectors were executed.

Table 1 presents the results of the simulation sessions for each of the ten Booth multipliers. The columns provide information about the number of clocks required by each multiplier type, the period ($\omega$), the delay ($\Delta$) for calculating the product, and the maximum frequency (f) of operation. In addition, the frequency of the Product Select multiplier is compared to that of the conventional multiplier for the same radix.

Table 1: Multiplier Timing

| Multiplier Type | # clks | $\omega$ (ns) | $\Delta$ (ns) | f (MHz) | % gain |
|---|---|---|---|---|---|
| Radix 2 | | | | | |
| Conventional | 16 | 16 | 256 | 62.50 | |
| Prod. Select | 16 | 13 | 208 | 76.92 | 23.08 |
| Radix 4 | | | | | |
| Conventional | 8 | 16 | 128 | 62.50 | |
| Prod. Select | 8 | 13 | 104 | 76.92 | 23.08 |
| Radix 8 | | | | | |
| Conventional | 6 | 17 | 102 | 58.82 | |
| Prod. Select | 6 | 14 | 84 | 71.43 | 21.43 |
| Radix 16 | | | | | |
| Conventional | 4 | 18 | 72 | 55.56 | |
| Prod. Select | 4 | 15 | 60 | 66.67 | 20.00 |
| Radix 32 | | | | | |
| Conventional | 4 | 20 | 80 | 50.00 | |
| Prod. Select | 4 | 16 | 64 | 62.50 | 25.00 |

One would expect the difference in period between the regular and product select multiplier to correspond to the delay time of passing through the control circuitry. This is true except for the cases where the multiplexer delay increases between the regular and modified versions. This is seen in the Radix 16 case where the delay to generate the select signals is 4 ns; however, the multiplexer adds another level of gates to allow the P value to be selected in the multiplexer of the modified version. Thus we see only a 3 ns gain instead of 4 ns. This also explains why the Radix 32 actually sees the greatest percent speedup. It also has a 4 ns propagation time through its control logic; however, the size of its multiplexer does not increase when the product select technique is used. Therefore it gains 4 ns instead of 3 ns. It is interesting to note, though, that the Radix 32 multiplier is actually a very bad choice for 16 bit multiplication. Like the Radix 16 multiplier, it still requires four clock cycles to complete its operation; however, each clock cycle actually takes longer. In addition, as will be shown later, it also takes up much more space. It is therefore important to consider the size of the operands before selecting the radix of a multiplier.

Another consideration for selecting a multiplier radix is the cost in terms of number of gates. Although increasing the radix of a multiplier decreases the number of clock cycles required to calculate the results, a penalty is paid in gates. The product select circuitry also requires extra gates. Table 2 compares the number of gates required by the conventional multipliers at various radices against the product select multipliers of the same radix.

Table 2: Product Select Multiplier Gate Count

| Multiplier Type | gates | flip flops | total gates | % gain |
|---|---|---|---|---|
| Radix 2 | | | | |
| Conventional | 340 | 70 | 1180 | |
| Prod. Select | 554 | 70 | 1394 | 18.14 |
| Radix 4 | | | | |
| Conventional | 409 | 70 | 1249 | |
| Prod. Select | 1064 | 70 | 1904 | 52.44 |
| Radix 8 | | | | |
| Conventional | 811 | 73 | 1687 | |
| Prod. Select | 2363 | 73 | 3239 | 92.00 |
| Radix 16 | | | | |
| Conventional | 1875 | 71 | 2727 | |
| Prod. Select | 5453 | 71 | 6305 | 131.21 |
| Radix 32 | | | | |
| Conventional | 4370 | 76 | 5282 | |
| Prod. Select | 12012 | 76 | 12924 | 144.68 |

Note that the difference in area between a conventional Radix 2 multiplier and a conventional Radix 4 multiplier is only 69 gates while the performance doubles. As the radix increases, the number of adders required causes the gate count to increase very rapidly while the number of clock cycles required does not decrease as quickly.

The gate count is much higher for the product select multipliers; however, at lower radices, their use can still be warranted if space permits.

It can be seen that the Radix 2 multiplier only increases in area by 18% while improving its performance by 23%. The Radix 4 multiplier gate count increases by just over 50% and also has a performance improvement of 23%. For the five multiplier types evaluated here, the performance increase remains between 20 and 25% while the percent increase in gate count increases with the radix. Note that the percent increase begins to level off at Radix 16. This is because the higher radix multipliers already require so many adders to generate the various multiples of R that the addition of the adders required to generate the potential partial products does not have nearly as great an effect.

Tables 1 and 2 compare the conventional and product select multipliers according to performance improvement and area gain at each radix. By comparing the ratio of these two values, the product select technique can be analyzed to determine its suitability in a given application over the conventional multiplier at the corresponding radix. This comparison is made in Figure 3.

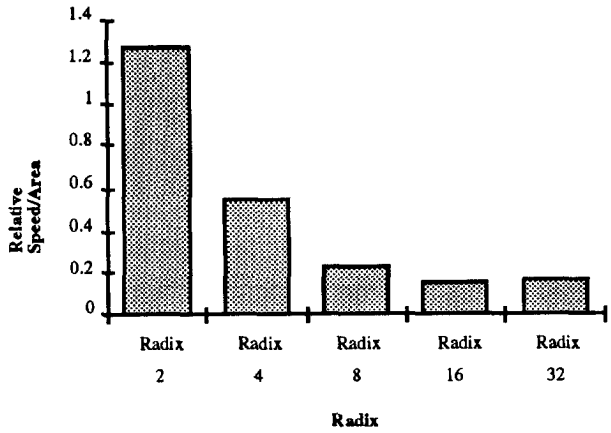


Figure 3: A Comparison of Relative Speed and Area

The Radix 2 multiplier actually demonstrates a greater speedup than increase in area. The Radix 4 multiplier has nearly a 2:1 ratio of area to speedup. The higher radix multipliers begin to increase gate count rapidly as the product select technique is applied. Notice that the Radix 32 ratio is actually more favorable than the Radix 16 ratio. This is because of the fact that the Radix 32 multiplier multiplexer does not incur any greater delay in the product select case than in the regular case. This allows a 4 ns decrease in cycle time. For numbers much larger than 16 bits, this could indicate that the Radix 32 multiplier might be a good choice over the Radix 16 multiplier; however, as discussed earlier, it is not a good choice for 16 bit numbers. In general, we should expect to see a similar "dip" each time the radix increases past a number that is a power of four, provided the overall cycle times for the two radices are not too dissimilar. Past that power of four, the multiplexer does not increase in size between the regular and product select versions. At the power of four radix, the one gate delay difference between the multiplexers would allow the area/performance ratio to favor the higher radix.

One other analysis can be made in determining which multiplier to use. Table 3 compares each of the multipliers to the standard Radix-2 multiplier.

Table 3: Comparison of Multipliers to Standard Radix 2 Multiplier

| Multiplier Type | speedup vs. Radix 2 | gate count vs. Radix 2 |
|---|---|---|
| Radix 2 | | |
| Conventional | – | – |
| Prod. Select | 23.08% | 18.14% |
| Radix 4 | | |
| Conventional | 100.00% | 5.85% |
| Prod. Select | 146.15% | 61.36% |
| Radix 8 | | |
| Conventional | 60.16% | 42.97% |
| Prod. Select | 67.19% | 174.49% |
| Radix 16 | | |
| Conventional | 255.56% | 131.10% |
| Prod. Select | 326.67% | 434.32% |
| Radix 32 | | |
| Conventional | 220.00% | 347.63% |
| Prod. Select | 300.00% | 995.25% |

In an overall comparison to the Radix 2 Booth multiplier, both the Radix 2 and Radix 4 Product Select Booth multipliers provide an increase in performance that is greater than the increase in gate count. A Radix 4 Product Select Booth multiplier represents a 61% increase in gates over a Radix 2 Booth multiplier yet decreases overall latency by 152 ns. This would correspond to Radix 2 operation at 153.85 MHz, an improvement of 146.15% over its normal 62.5 MHz. In this comparison, the Radix 16 multiplier provides a 326% gain over the Radix 2 multiplier at a cost of 434.3% more area. Figure 4 is similar to Figure 3 which analyzes the ratio of area gain and speedup; however, Figure 4 compares the product select multipliers against the conventional Radix 2 multiplier.

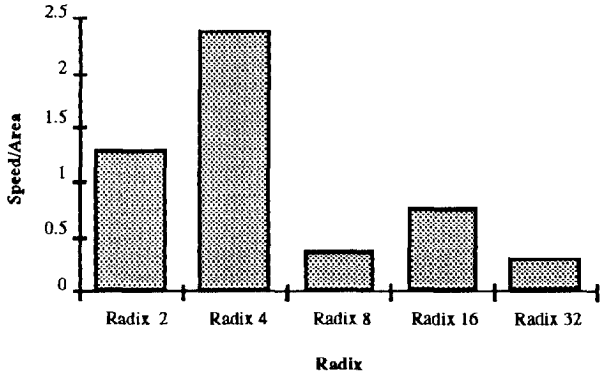


Figure 4: Comparison of Area and Speed vs. Conventional Radix 2

From Figure 4 it is evident that the radix 2, 4, and 16 product select multipliers maintain an acceptable ratio of performance gain to area gain. If performance is the driving issue, a Radix 16 product select multiplier provides over a 4x speedup at roughly a 5x increase in gate count.

## 4 Conclusions

When selecting a multiplier, many factors influence the decision. The available area and the performance desired are the two major factors and are generally at odds with each other. As performance increases, so does the area required for the multiplier. It is left to the designer to weigh the tradeoffs. If area considerations are important but can be relaxed enough to get a 2.5x increase in performance, the Radix 4 product select multiplier could be chosen over the Radix 2 multiplier. Given large word sizes, a higher radix means higher performance in terms of latency. Practically, however, the number of bits also plays a large role in determining the radix of the multiplier as was shown by the fact that the Radix 16 multiplier outperforms the Radix 32 multiplier for 16 bit operands. In addition, the slower clock rates of the higher radix multipliers might make them less attractive for pipelined applications.

Further study of the product select multiplier should be directed towards the study of higher radices and towards further optimizing the generation of the various multiples of R. The former becomes more practical as standard word sizes for contemporary computers increase while the latter should always be a consideration so that the setup time for the multiplier is not the driving factor in determining its performance.

## References

[1] Andrew D. Booth, "A Signed Binary Multiplication Technique," Quarterly Journal of Mechanics and Applied Mathematics, Vol. 4, pp. 236-240, 1951.

[2] O. L. MacSorley, "High-Speed Arithmetic in Binary Computers," IRE Proceedings Vol. 49, pp. 67-91, 1961.

[3] Homayoon Sam and Arupratan Gupta, "A Generalized Multibit Recoding of Two's Complement Binary Numbers and Its Proof with Application in Multiplier Implementations," IEEE Transactions on Computers, Vol. 39, No. 8, pp. 1006-1015, August, 1990.

[4] O. J. Bedrij, "Carry-Select Adder," IRE Transactions on Electronic Computers, pp. 340-346, 1962.

[5] Earl E. Swartzlander, Jr., Computer Arithmetic Vol. I, IEEE Computer Society Press, Los Alamitos, CA, pp. 4.9-10, 1990.