Technology Scaling Effects on Multipliers

Hesham Al-Twaijry and Michael Flynn

Technical Report : CSL-TR-96-698

July 1996

This work was supported using facilities supported by NASA contract NAG2-842 and a fellowship from Saudi Arabia.

Technology Scaling Effects on Multipliers

by

Hesham Al-Twaijry and Michael Flynn

Technical Report : CSL-TR-96-698

July 1996

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305-9040
pubs@shasta.stanford.edu

Abstract

Booth encoding is a method of reducing the number of summands required to produce the multiplication result. This paper compares the performance/area tradeoffs for the different Booth algorithms when trees are used as the summation network. This paper shows that the simple non-Booth algorithm is not an efficient design, and that for small feature sizes the performance for the different Booth encoding schemes are comparable in terms of delay.

The report also quantifies the effects of wires on the multiplier. As the feature size continues to decrease wires will provide an ever increasing portion of the total delay, Booth 3 becomes more attractive since it is smaller.

Key Words and Phrases: Floating-Point, Multiplication, Booth, Trees, Technology scaling

Copyright © 1996
by
Hesham Al-Twaijry and Michael Flynn

Contents

List of Figures

List of Tables

1 Introduction

Multiplication is one of the basic arithmetic operations. The speed of the multiplier is a critical issue in determining the performance of the microprocessors. The advent of VLSI has given chip designers the ability to integrate multipliers as part of microprocessors. Therefore, it has become common for modern microprocessor to have floating point multipliers implemented fully in hardware, in contrast to in software as in the previous generations. In fact the very latest processors also implement integer multiplication in hardware. Integer multiplication is used to speed up address translation, array indexing, etc.

Multiplication is the process of adding a number (summand) a certain number of times. Each summand is a chosen multiple of one of the operands (multiplicand), based upon the value of certain bits of the other operand (multiplier). These summands need to be added together to produce the final result. The addition of each pair is a relatively long latency carry-propagate addition. In order to reduce the total time required to produce the final result, a redundant form of addition, most commonly *carry-save* addition, is used. In carry-save addition the summands are split into columns. Each bit of the summands is called a partial product. The columns of partial products are added independently from adjacent columns. The addition of the partial products in each column is achieved by the use of *counters*. These counters are circuits which encode the number of ones in their inputs. The counters can be connected in several ways.

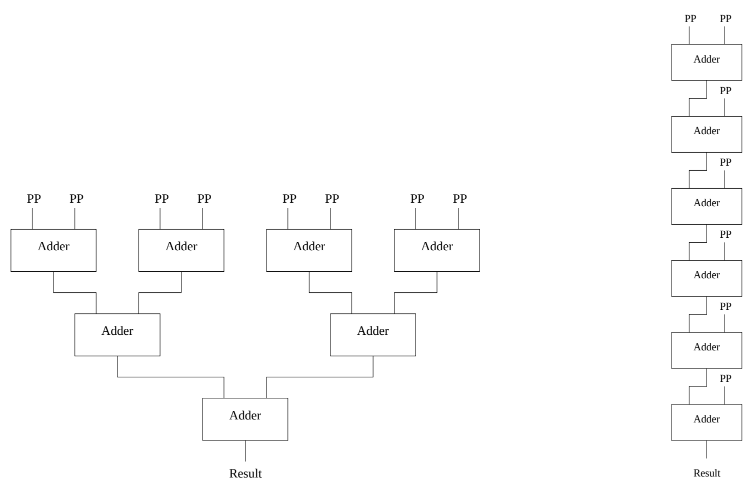


Figure 1: Counter interconnection Schemes (a) Parallel (b) Linear

In high speed multipliers, the counters are connected in parallel using tree structures as shown in figure 1(a), in contrast to serially as in linear arrays. The number of adders needed to reduce the partial products is the same for both trees and arrays. The only difference is that trees have more complex interconnections. The more complex interconnection affect the design of the multiplier when it is part of a chip because of the limited number of wires available that are used for interconnecting the counters.

The number of summands that must be added to give the multiplications' result can be reduced by using Booth encoding [2]. In Booth encoding the number of summands is reduced by recording the multiplier bits into groups that select multiples of the multiplicand. Higher order Booth encoding reduces the number of summands by a greater degree by encoding larger groups of multiplier bits and therefore requiring a larger group of multiples to select from, and consequently a more complex selection table.

Sign Bit (1)	Normalized Fraction (52)	Biased Exponent (11)

Figure 2: Floating Point Number Standard

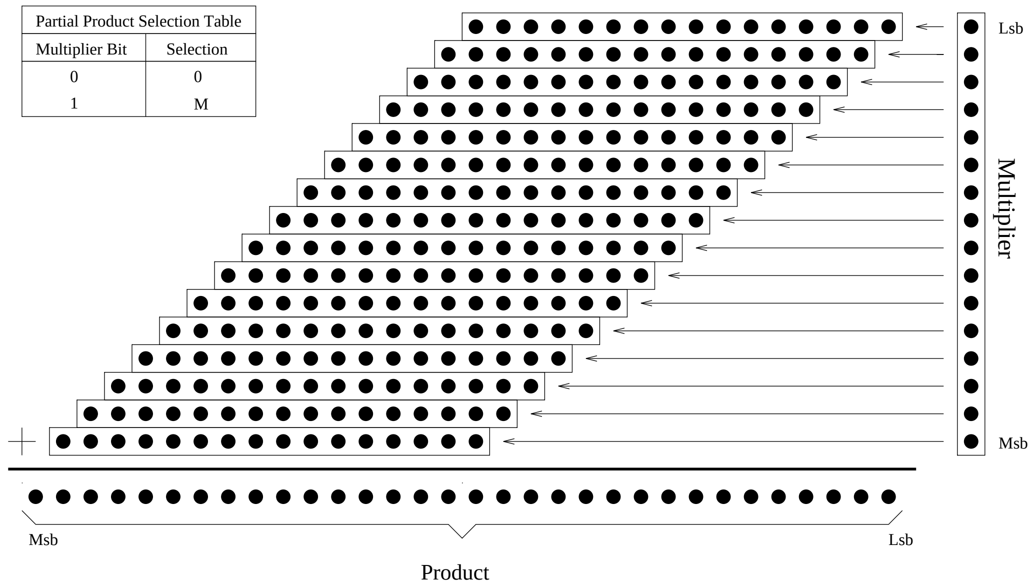
The multiplier under consideration uses the IEEE floating point arithmetic standard [1]. The format for double precision numbers, as defined by the standard, is shown in figure 2. The standard defines numbers in a sign-magnitude, normalized format. The standard has a normalized significand, that is the most significant bit of the fraction is always 1, and therefore is not stored. The significand effectively becomes 53 bits. To achieve the rounding accuracy defined by the standard, the full 106 bit result has to be calculated, even though almost half of it is used only for rounding.

The multiplier is part of a processor's datapath which forces the width of each subcell or *bit pitch* to be constant. The required structure for connecting the counters for each topology is achieved by varying the interconnection network of the adders. The interconnection network is routed on top of the adders themselves. Based upon the number of wiring tracks available and the number of tracks required by the chosen topology, one uses circuits that propagate the signal or circuits that produce both the signal and its complement.

This study investigates the relationship between the topology of the partial product interconnections and the encoding scheme used. It also studies the effect of these topologies and encoding schemes on the latency and area of the multiplier as the feature size decrease, when the multiplier is part of a larger system. This study also quantifies the effects of wires on the multiplication process.

2 Encoding Methods

There are several methods for encoding the summands. A popular method when large multipliers were built using MSI parts, was the use of smaller multipliers to generate the partial products. These smaller multipliers were simply the truth table representation of the multipliers output and where built out of ROMs. With the advent of VLSI and the requirement to build multipliers as part of a single chip, this method is no longer used. Booth encoding is the most common method used to reduce the number of summands.

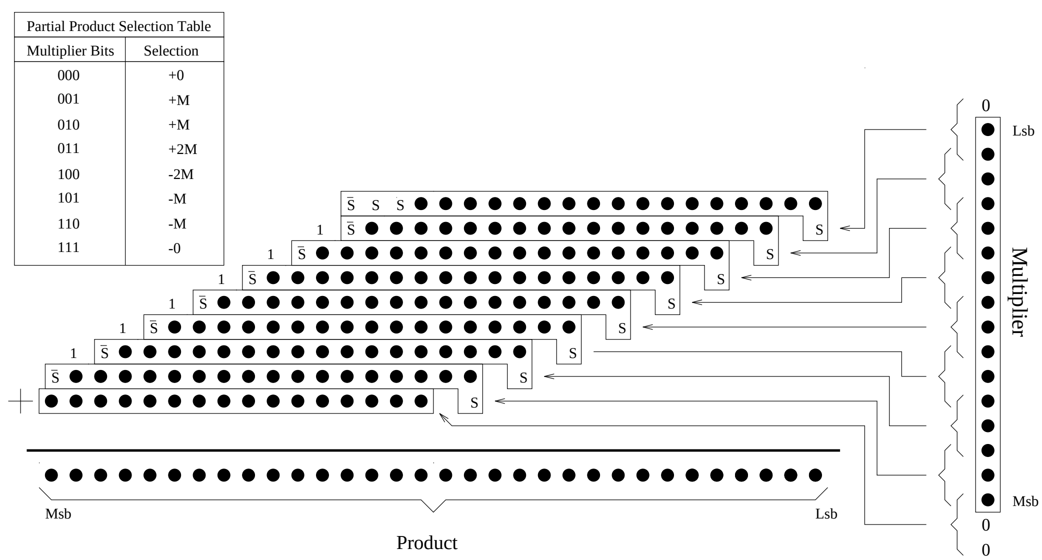


Partial Product Selection Table	
Multiplier Bit	Selection
0	0
1	M

Figure 3: Non-Booth Encoding

2.1 Non-Booth

The first and simplest method for encoding is non-Booth as shown in figure 3. This algorithm is simply a shift and add algorithm where the multiplicand is conditionally added to produce the final result. In this algorithm the summand is selected from the set {0, M}. This algorithm's summand selection logic is a simple AND gate. Unfortunately, there is no reduction in the number of summands that need to be summed to produce the final result.



Partial Product Selection Table	
Multiplier Bits	Selection
000	+0
001	+M
010	+M
011	+2M
100	-2M
101	-M
110	-M
111	-0

Figure 4: Booth 2 Encoding

2.2 Booth 2

The Booth algorithm attempts to reduce the number of the summands by recoding the multiplier so that groups of its bits select multiples of the multiplicand. The Booth algorithm as it was originally proposed performed the encoding serially. Therefore, the Modified Booth Algorithm [2] which performs the encoding in parallel is used. In this algorithm, the multiplier is partitioned into overlapping groups of 3 bits. Each group is decoded in parallel to select a multiple of the multiplicand from the set $\{\pm 2M, \pm M, \pm 0\}$, as shown in figure 4. All of these multiples are obtainable by simple shifting and complementation. The final result of the multiplication is obtained by adding the summands together. Negative multiples, in 2's complement form, are obtained by performing bit by bit complementation of the corresponding positive multiple, and adding a 1 at the least significant position (the S bits along the right side of the parallelogram). For the multiplication to produce the correct result, each summand needs to be sign extended. Fortunately, the sign extension that is required is minimal as shown in figure 4. A rigorous proof that this is a sufficient sign extension is presented in [3] .

Using this algorithm; the number of summands is $\lceil \frac{n+1}{2} \rceil$. The extra 1 in the expression comes from the need to ensure that the last summand is a positive multiple of the multiplicand. This is achieved by adding at least an extra zero to the left of the *multiplier.* It should be pointed out that Booth 2 does not reduce the area by almost half compared to non-Booth. The area reduction is less than that because of the more complex summand selection logic (Booth encoders) and partial product generators (Booth muxes).

Partial Product Selection Table			
Multiplier Bits	Selection	Multiplier Bits	Selection
0000	+0	1000	-4M
0001	+M	1001	-3M
0010	+M	1010	-3M
0011	+2M	1011	-2M
0100	+2M	1100	-2M
0101	+3M	1101	-M
0110	+3M	1110	-M
0111	+4M	1111	-0

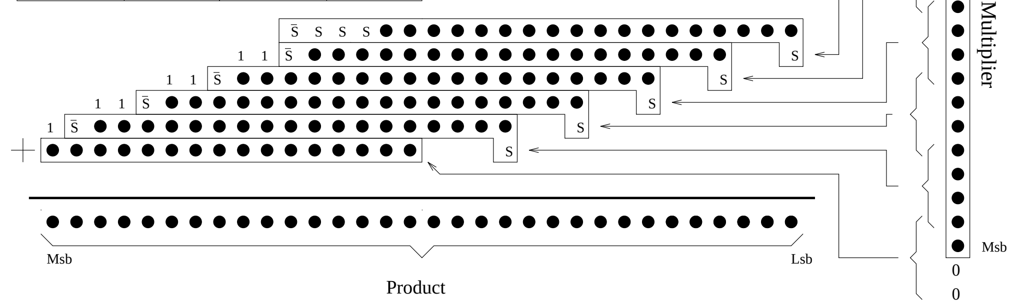


Figure 5: Booth 3 Encoding

2.3 Booth 3

It is also possible to reduce the number of summands required to produce the final result using shift amounts greater than two. In Booth 3 the multiplier is partitioned into overlapping groups of 4 bits, each group is decoded in parallel to select a certain summand. Each summand could be from the set $\{\pm 4M, \pm 3M, \pm 2M, \pm M, \pm 0\}$, as shown in figure 5. All the multiples except 3M are easily obtainable, by simple shifting and complementing.

The generation of the 3M multiple, referred to as a hard multiple, can not be obtained by simple shifting and complementation. Rather, a full carry propagate addition is required. This carry propagate adder increases the latency of the Booth 3 multiplier. In some processors the generation of the 3M multiple is designed as a separate cycle in the processor's pipeline. The 3M adder is a special purpose adder that is designed to add (M + 2M). This allows for some circuit optimizations which make it faster than a regular adder. This algorithm also has the most complex selection logic. Using this algorithm the number of summands is $\lceil \frac{n+1}{3} \rceil$. The extra 1 in the expression comes from the need to ensure that the last summand is a positive multiple of the multiplicand. This is an unsigned operand.

Partial Product Selection Table

Multiplier Bits	Selection	Multiplier Bits	Selection
0000	K+0	1000	K-4M
0001	K+M	1001	K-3M
0010	K+M	1010	K-3M
0011	K+2M	1011	K-2M
0100	K+2M	1100	K-2M
0101	K+3M	1101	K-M
0110	K+3M	1110	K-M
0111	K+4M	1111	K-0

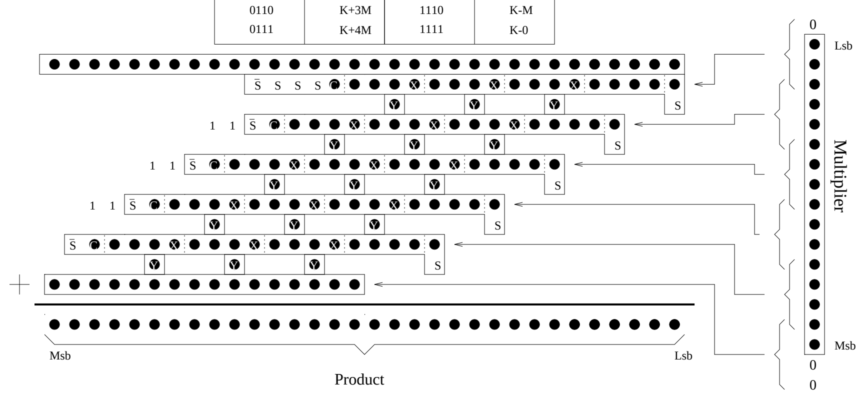


Figure 6: Redundant Booth 3 Encoding

2.4 Redundant Booth 3

Redundant Booth 3 aims to eliminate the need for the hard multiple generation by generating an equivalent partially redundant representation [4]. The partially redundant representation is achieved by using a series of smaller adders, with no carry propagation between the adders, as shown in figure 6. If the adders are small enough, then the carries are not propagated across many bits and therefore the generation of the three times multiple should not take a long time. The adders are designed to take as much time to produce their results as the Booth encoding process takes to decode the bits of the multiplier and select the appropriate multiplicand multiple as the summand. A constant is added to each summand so that the positive and negative multiples of the multiplier will have zeroes in the *empty* gaps between the carries. This will allow these locations to be ignored, i.e. there is no need to add the zeroes so no counters are needed for these locations. This constant is required to be the same number for all the possible multiples of the multiplicand. The negative of the sum of all the constants is added to the summands at design time so that the net result added to the tree is zero. The negative of the sum of the constants is called the compensation constant. It is possible to add the compensation constant at design time because each of the summands constant is independent of the multiple chosen.

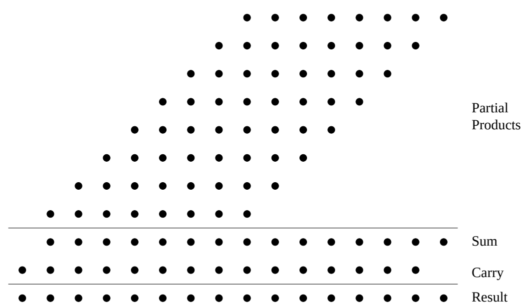


Figure 7: Multiplier's shape

3 Topology

The different encoding schemes all produce a parallelogram shaped multiplier structure as shown in figure 7. The rows of the parallelogram are the summands which are added together to produce the final result. In order to reduce the final result, the time consuming carry-propagate addition is performed once. This is acheived by using a redundant form of addition, called carry-save addition. In carry-save addition carries only propagate to the next column. Therefore, each column in the parallelogram is treated seperately.

The partial products in each column are reduced to two bits, which are part of the sum and carry vectors. The reduction of the partial products is acheived by using counters. Counters are circuits that encode the number of ones in their inputs. There are many ways to connect the counters to produce the final result. These different ways for interconnecting the counters are called topologies. the number of counters needed to reduce the partial products is the same for all the topologies. The topologies differ in the compexity of the interconnections used to connect the counters. The topologies that connect more partial products in parallel have more complex interconnections.

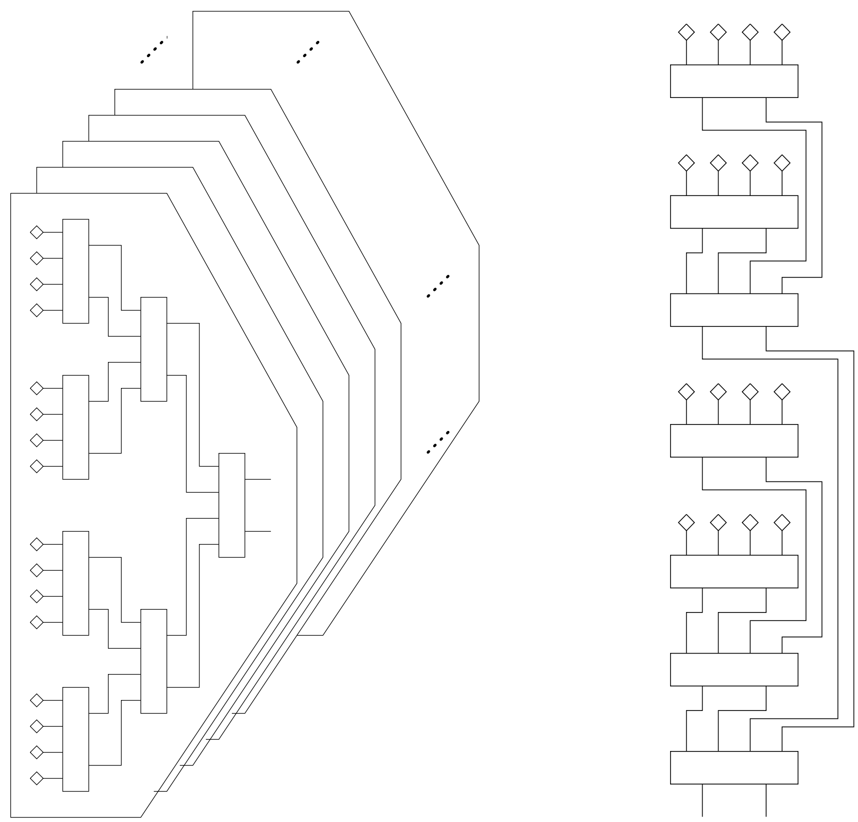


Figure 8: (a) Counter Placement (b) Actual Layout in Bit Pitch

The counters used in the reduction of the partial products are laid out in rows. Each row reduces its inputs by the compression ratio of its counters. This organization of the counters forms a two dimensional plane. The plane is repeated for each column of the parallelogram. This creates a three dimensional structure as shown in figure 8a. However, chips are designed as a planar process. Therfore, each of these planes has to be flattened. The flattening process is acheived by placing the counters in a linear fashion as shown in figure 8b. The width of partial product reduction structure is the width of one counter. In contrast to the two dimensional structure, the counters outputs when laid out linearly are inputs for non-adjacent counters. The outputs of the counters will therefore have to bypass the intermediate counters that lay between the inputs and outputs of the counters. The bypassing is acheived by routing wires to interconnect the counters. The wires are routed over the intermediate counters. The width of the partial product reduction structure is the width of a counter. This width is the bit pitch for the cells in the datapath. Each bit pitch has a limited number of wires available for routing the signals. Therefore, the more complex topologies may require more wires than available and are thereby impossible to route.

There are many different topologies that differ in the number of wires that each topology requires and in the number of counters used. These topologies include:

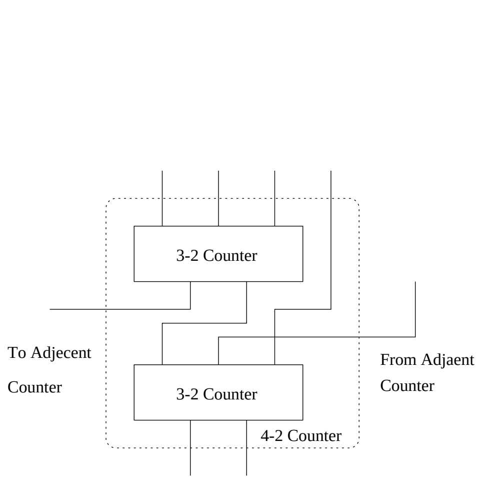


Figure 9: (a) "4-2" Counter Logical Construction (b) Double Linear Array

3.1 Double Linear Array

The double linear array is constructed using "4-2" counters. The 4-2 counter is logically constructed from two 3-2 counters as shown in figure 9a. However, actual implementations of the "4-2" counter are constructed in such a way that the total delay is less than that for two serial connected "3-2" counters. The 4-2 counter is symmetric in that it has a 2 : 1 reduction ratio, while the 3-2 counter is not symmetric.

Linear arrays are the simplest topology in which the counters are serially connected. In the linear array the delay from each of the inputs is proportional to the location at which it is added to the array. The double linear array reduces the delay of a simple array in half by connecting the even numbered counters and the odd numbered counters separately as shown in figure 9b. The two counter chains are combined by the last counter to produce the final result. The number of counters in the critical path is linearly proportional to the number of partial products being reduced. This is in contrast to trees that usually are logarithmically proportional to the number of partial products. The number of wires that are needed to connect the counters in an array structure is constant and independent of the number of partial products being reduced.

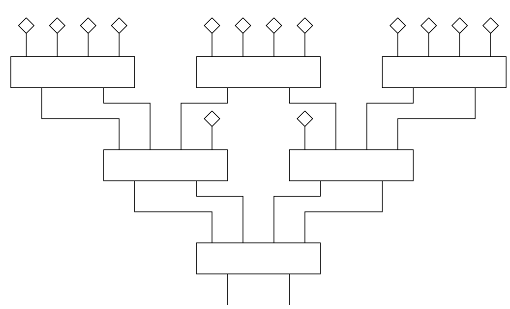


Figure 10: Binary Tree

3.2 Binary Tree

The 4-2 counter tree [5] has a regular and symmetric structure, as shown in figure 10. In the 4-2 tree, for every four inputs taken at one level, two results are produced at the next level. This can be thought of as a redundant binary tree, since every counter reduces two redundant numbers into one redundant number. The 4-2 tree's binary nature makes it commonly used in pipelined, and iterative multipliers. The symmetric nature of the 4-2 counter facilitates the addition of latches that are needed for pipelining after each 4-2 counter. Iterative and pipelined 4-2 counter trees use the same structure for each bit pitch with the output of previous stages being fed back after the appropriate shifting. The number of counters in the critical path for the binary tree is a logarithmic function of the number of partial products that need to be reduced.

The 4-2 counters do not have a constant requirement for wires. The number of wires needed increases by two when the number of partial products is doubled. Their wiring requirement is similar to that of Wallace trees [7], in that they are both logarithmic. However, the growth rate of wiring tracks in 4-2 trees is smaller. Also, their wiring requirement is more regular, since Wallace trees, which use 3-2 counters, are extremely irregular making them notoriously difficult to layout.

The advantage of the binary tree reduction of the 4-2 trees is not all that significant for IEEE double precision numbers since the significand size is not a power of two.

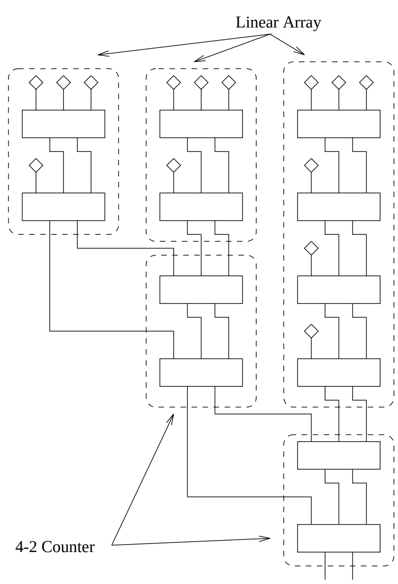


Figure 11: Balanced Delay Tree

3.3 Balanced Delay Tree

The balanced delay tree was proposed by Zuras and McAllister [6]. The ZM tree is based upon the idea of balanced delay chains of counters. Trees are constructed by combining progressively longer serial chains into serial chains below them. The connection between the two chains is made when the total delay of the upper chain is equal to the delay of the lower chain. The connection is made when the number of counters in the critical path of upper chain of counters is as long as the delay of the critical path of the chain of lower counters. The ZM tree is "balanced" in the sense that the delays of all chains are approximately equal and that the wire usage is balanced between all counters. The previous method builds ZM trees of type one, which require only two wires to feed the output of one counter to the input of a non-adjacent counter, as shown in figure 11.

This tree structure has a very regular layout and it requires only a few primitive cells. This type of tree generally uses more levels of counter delay than the Wallace tree. To reduce the number of levels, higher order ZM trees are constructed, by iteratively replacing the largest chains with ZM trees of type 1. These higher order trees require a larger number of tracks, and are less regular. The number of tracks required by ZM trees is $2P$, where P is the order of the tree, and the number of levels is $O(N^{\frac{1}{P+1}})$.

ZM trees are not easily pipelined. The pipelining of a ZM tree requires that the outputs of the Booth muxes that are not at the first level, to be latched in addition to the outputs of the 3-2 counters. The number of latches required is therefore greater than the number of latches in a 4-2 counter tree. ZM trees can be built to produce the result iteratively using a structure that is similar to a 4-2 tree.

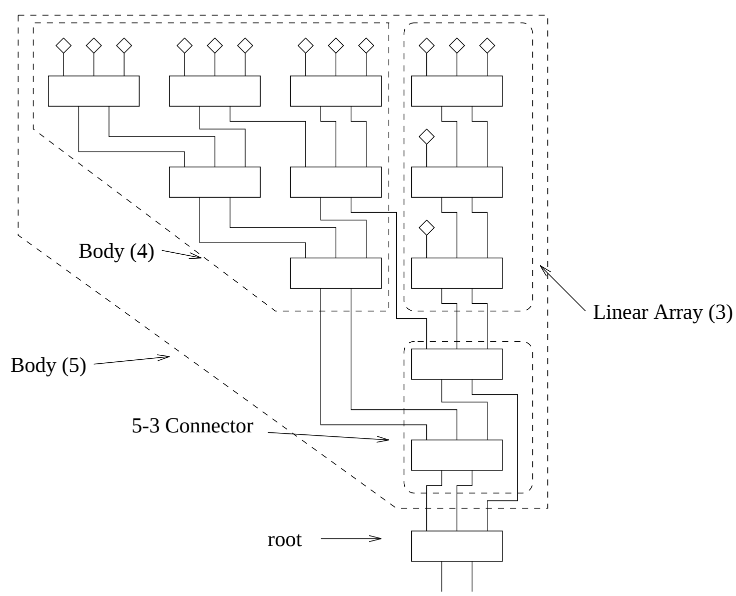


Figure 12: Overturned Staircase Tree

3.4 Overtuned Staircase Tree

The Overturned Staircase Tree was proposed by Mou and Jutand [8]. It is called an overturned staircase because the way the counters are connected resembles a staircase. This method divides a tree into a body and a root. The root is the last 3-2 counter in the tree. The body is constructed recursively. A body of height k, where k is the number of 3-2 counters in the critical path, is constructed from a body of height k-1 and a linear array of height k-2. The linear array and the body are joined using a 5-3 counter. The 5-3 counter is constructed from two 3-2 counters in series. This method build OS trees of type one, as shown in figure 12. This tree structure requires a few primitive cells. It requires 3 tracks to route signals between non-adjacent counters. The OS tree uses more wiring tracks than the ZM tree. The OS tree needs more primitive cells and it has a less regular structure, compared to the ZM tree.

OS tree structure can give the minimum number of counter levels for most numbers of partial products. However, to achieve this, one has to use higher order OS trees. Higher order OS trees can be built by replacing the linear arrays with OS trees of type one. The higher order trees require more wiring tracks. The number of tracks required by OS trees are *3P* , where P is the order of the tree, and the number of levels is $O(N^{\frac{1}{P+1}})$.

OS trees are not easily pipelined. The pipelining of a OS tree requires that the outputs of the Booth muxes that are not at the first level, is be latched in addition to the outputs of the 3-2 counters. Thus, as in ZM trees, the number of latches required is greater than the number of latches in a 4-2 counter tree. OS trees can be built to produce the result iteratively using structure that is similar to ZM tree. However, OS trees are not typically used for iterative multipliers, since 4-2 trees give a more regular topology, using the same number of counter levels.

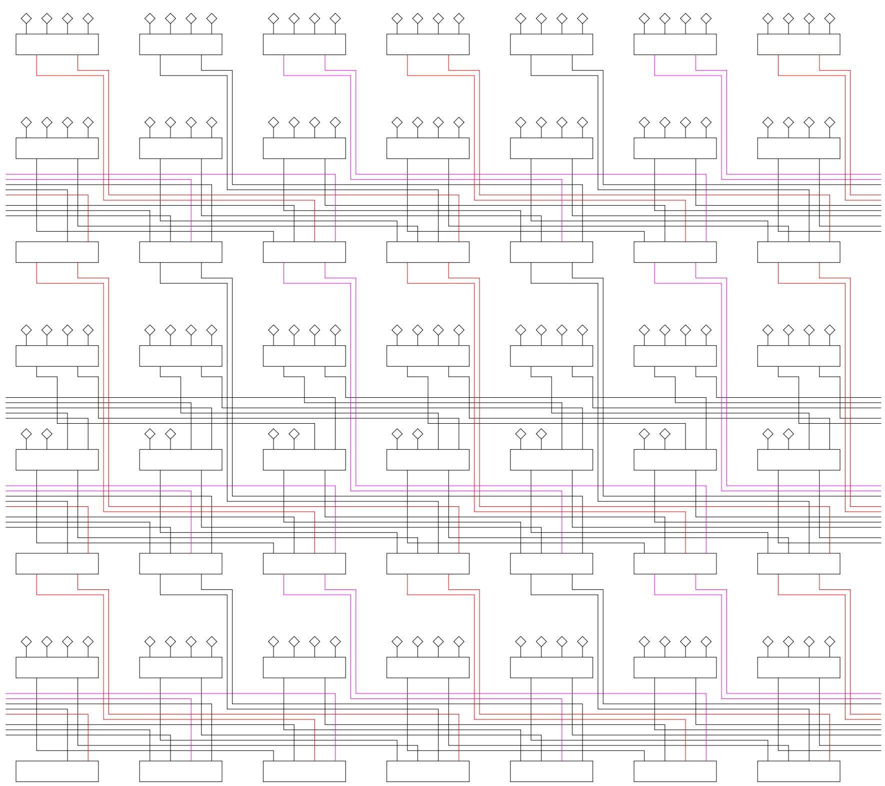


Figure 13: Booth 3 Higher Order Array

3.5 Higher Order Arrays

Higher Order Arrays is a class in which the counters are designed as several linear array chains, as can be seen in figure 13. The chains are combined in parallel when the delay of the upper chain is equal to the delay of the lower chain. This class of arrays can in fact be thought of as a collection of ZM trees of type one. The ZM trees have been designed for the column with the largest number of inputs. This design is replicated for all other columns. In this design, the non-critical columns are not optimized. This design trades the performance of the non-critical columns for regularity, as shown in figure 13. The regularity of the higher order tree is proportional to the number of linear arrays that are combined. The smaller the number of arrays the more regular the design.

Higher order trees can be classified according to the lengths of the chains of partial products before the combining occurs. For example, the 6-6-8-8 array has a linear array that combines 6 partial products which is combined with an array the combines 6 partial products. The resulting structure is then combined with an array that combines 8 partial products. Finally, the resulting structure is combined with an array that sums 8 partial products.

Higher order arrays are pipelined as easily as arrays. However, since their design is proposed to reduce the latency of the multiplier using the smallest number of wiring tracks available, pipelined iterative higher order trees are not very attractive.

Since Higher order arrays are just ZM trees of type 1; they require only two tracks and there summing time is $O(\sqrt{N})$

3.6 Topology differences

The word topology refers to how the counters are connected. The different topologies each require a different number of wiring tracks that will be used for bypassing the intermediate counters.

Tree topologies give structures that have a small number of serial counters at the expense of complex interconnections. When the multiplier is part of a larger system, the number of available wiring tracks that can be used for bypassing is limited. Table 1 gives the minimum number of tracks that each topology requires to be routable.

Topology	Number of Wiring Tracks		
	Non-Booth	Booth 2	Booth 3
Linear Array	6	6	6
Binary Tree	16	12	12
Balanced Delay Tree	10	10	10
Overturned-Stair Case Tree	12	12	12
Higher Order Arrays	6	6	6

Table 1: Minimum Number of Wiring Tracks

The table was obtained for IEEE double precision numbers. The table includes the number of wiring tracks required to route the inputs and outputs of the partial product generators and the counters. The partial product parallelogram for the tree topologies is folded to save silicon area. The maximum number of wiring tracks occur in the folded region. This is because their are two comparably sized trees that whare the available wiring track. The highly regular array topologies are not folded since all partial product columns have the same structure, so aligning the partial products and routing the intercounter wires diagonally is possible.

The topology used and consequently the number of wiring tracks used and the number of wiring tracks available have an effect on the choice of circuits used to implement the counters. The circuits used to build the counters can be broadly classified as belonging to either of two families; single-ended and complementary signal circuits. The complementary signal circuits are usually faster since they produce and propagate both the signal and its complement and no signal inversions are required.

4 Results

The transistor models that are used in the simulations are scalable. The transistors and wires are assumed to scale to what the electrical limits dictate [9]. The fabrication process is not assumed to limit what is achievable for the transistor. The physical limits for the scaling of transistors come about due to leakage currents in the transistors and reliability concerns for the transistor.

As the feature sizes decrease to sub-micron feature sizes, the supply voltages are also decreasing. For extremely short sub-micron channel lengths the supply voltage will be in the 1-2 volt range. Therefore, at these small channel lengths the use of NMOS pass transistor logic will no longer be feasible because of the threshold voltage drops. At these small feature sizes the logic families that use both NMOS and PMOS transistor will be the ones that operate correctly.

The ratios between the latencies of the different encoding schemes and topologies will differ with the changing of the feature size. The differences are a result of the different rates at which the capacitances and resistances of the transistors and wires scale. Wires will scale at a slower rate than the transistors. This means that at small feature sizes the wires will contribute a larger part of the total delay.

The simulations are based upon the scalable HSPICE transistor models developed by McFarland [9]. The delays are for 25^oC operating temperature. The latencies are measure from the 50 % Vdd points.

The bit pitch for the floating point datapath is assumed to be 110λ. This pitch is wide enough for 12 wires to pass over the cells that make up the datapath. The wires are used for the routing of the partial products and the counter outputs. It should be noted that non-Booth binary tree results have been included although it is not possible to route binary tree in the number of tracks available. They will need a wider bit pitch. The wider pitch is used for the area calculations.

The adder that is used to generate the three times multiple for the Booth 3 encoding scheme is a special purpose adder that adds M + 2M. The specification that the adder only performs this addition give a considerable saving in both latency and area.

The counters used in the multiplier are all DPL circuits [10]. If the topology requirements for wires is small, the fully DPL circuit in which both the signal and its complement are produced and propagated. These topologies, which use DPL gates as they were originally defined, are direct candidates for having the counter circuits directly replaced with the dynamic logic circuits. However, if the topology is wire intensive, and it does not allow the use of the DPL circuits with both the signal and its complement generated a *single* valued version of DPL is used. In this version, the true value for each signal is generated, and the inverted values, which are needed by the counter internally, are generated by local inversion within the counters.

4.1 Binary Tree

4.1.1 Wire Effects

This section describes the wire effects for each of the encoding schemes when the topology is the binary tree, using 4-2 counters.

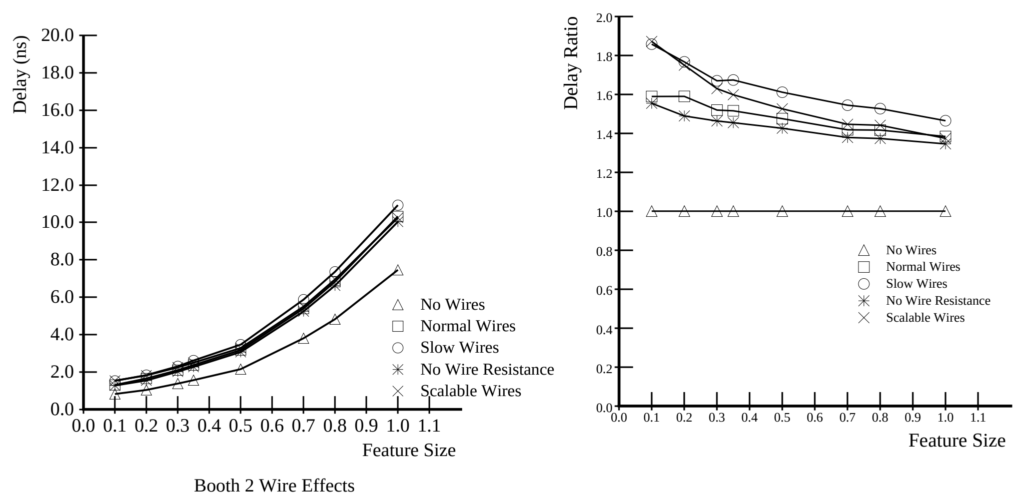


Figure 14: Booth 2: Overall 42 Tree (a) Delays (b)Ratio

Figure 14 gives the effects of wires on the total delay for Booth 2 encoding. From this figure it is apparent that the effects of the wires' capacitance are greater at smaller feature sizes. This is due to the wires capacitance providing a greater portion of the total capacitance for the circuit at the smaller feature sizes. The wire capacitance is proportional to both the area and length of the wires, while the capacitance of the transistors is proportional only to the areas. Therefore, the capacitance of the wires decreases at a slower rate than the capacitance of the transistors. For extremely small feature sizes the wiring capacitance will become dominated by the fringing capacitance, which is proportional to the wire lengths. Wire capacitance effects are noticeable because wire capacitance is approximately 40 % of the total capacitance.

Surprisingly, the wire resistance does not cause a significant portion of the delay. Wire resistance is in the order of a few percent of the total resistance with most of the total resistance coming from the equivalent resistance for the transistors. In addition as the feature sizes decrease wire resistance increases slowly, while the resistance of the transistors also increase but at a slower rate. Therefore at smaller feature the wires resistance ratio to the total resistance increases slightly, although the increase is so small that its effects on total delay are minimal.

Wires have a greater effect for non-Booth multipliers, when compared to Booth 2. Non-Booth has two contradicting factors affecting the wire effects on the total delay. The first effect is that it has more levels of logic. For Binary trees this extra logic is a single row of "4-2" counters, less the logic needed to obtain the summands for Booth 2, Booth encoder and Booth mux. The second effect is the extra long wires needed to route the results of the row of counters. This extra row of wires requires some long wires, which causes the wire capacitance to be a more significant part of the total capacitance, as compared to Booth 2. The effect of the extra row of counters on the ratio of wiring capacitance to total capacitance is not large enough to offset the extra capacitance of the extra wires. This causes the wires to account for a larger portion of the total capacitance. For non-Booth multipliers at small feature sizes, more than 50% of the delay is due to wire capacitance compared to 30% for the larger feature sizes. Wiring resistance effects are still negligible sice wire resistance still remains a few percent of total resistance.

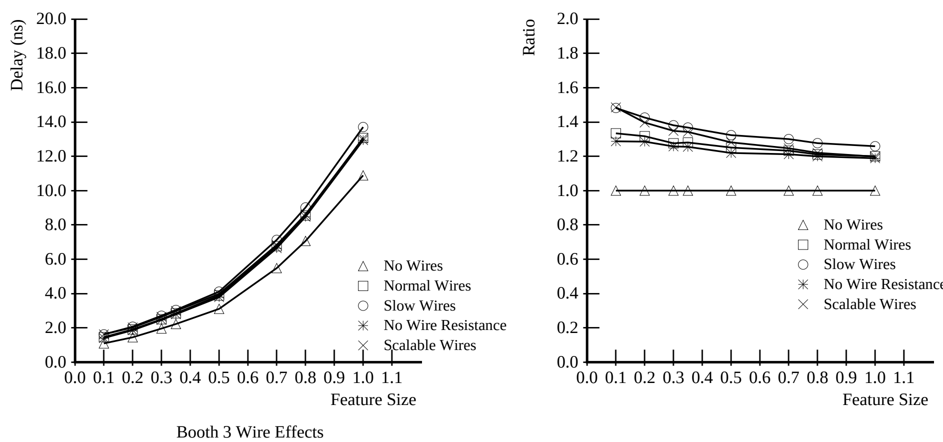


Figure 15: Booth 3: Overall 42 Tree (a) Delays (b)Ratio

Figure 15 gives the wire effects when Booth 3 is used as the encoding scheme. The wire capacitance's effect is not as significant for Booth 3 when compared to Booth 2. This is due to two factors. First, Booth 3 requires a greater number of logic levels to generate the three times multiple, in addition to the tree reduction. This large number of logic levels is equivalent to having a greater number of serial transistors in the critical path. This implies that the wiring capacitance is a smaller fraction of the total capacitance. So even if it decreases at a slower rate than the the active (transistor) capacitance, it still remains a smaller fraction of the total capacitance, when compared to Booth 2. Secondly, Booth 3 decreases the number of summands in the reduction tree by increasing the complexity of the selection logic. This reduction in the number of summands means that the wires are also shorter than Booth 2. This also has the effect that the wiring capacitance is a smaller portion of the total capacitance. Thus, the effects of the wire resistance for Booth 3 are even less than those for Booth 2.

The effects of wires for Redundant Booth 3 most closely match those for Booth3. The only differences are that redundant Booth 3 has longer wires and a slightly smaller number of logic levels. Therefore redundant Booth 3 is affected by wires to a slightly larger degree than Booth 3.

4.1.2 Scaling Effects

This section compares the different topologies for each different scaling scenario.

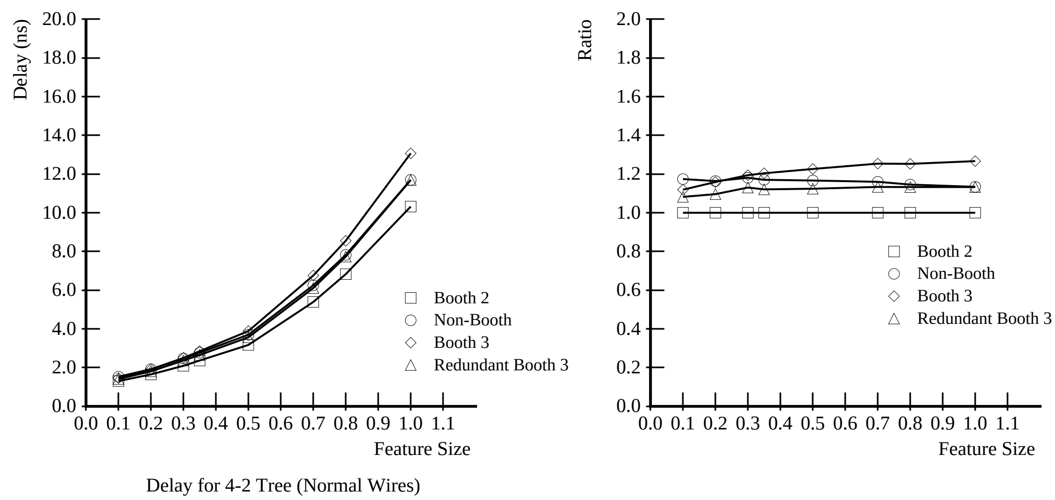


Figure 16: (a) Delays (b)Ratio

Figure 16 compares the different encoding schemes under the assumption that the wires will scale as predicted. This figure shows that Booth 2 always has the least latency. However, at very small feature sizes Booth 3 is only 10% slower while being significantly smaller.

If the wire resistance is negligible then what will happen will closely resemble what is expected to happen for normal scaling of wires. This means that the resistance of the wires has minimal effect on the circuits. This is because, although wire resistance will scale at a slower rate than the transistors equivalent resistance, the wire resistance will still remain an insignificant part of the total resistance.

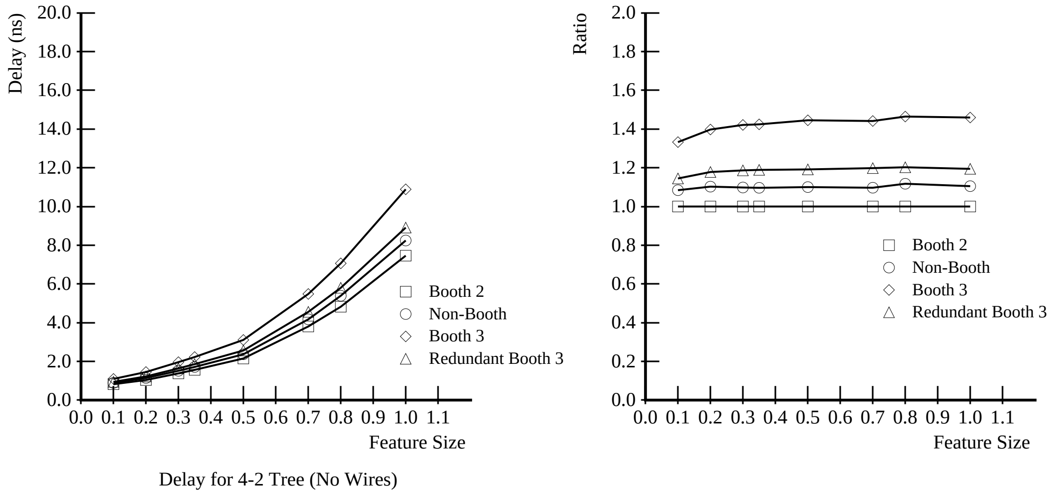
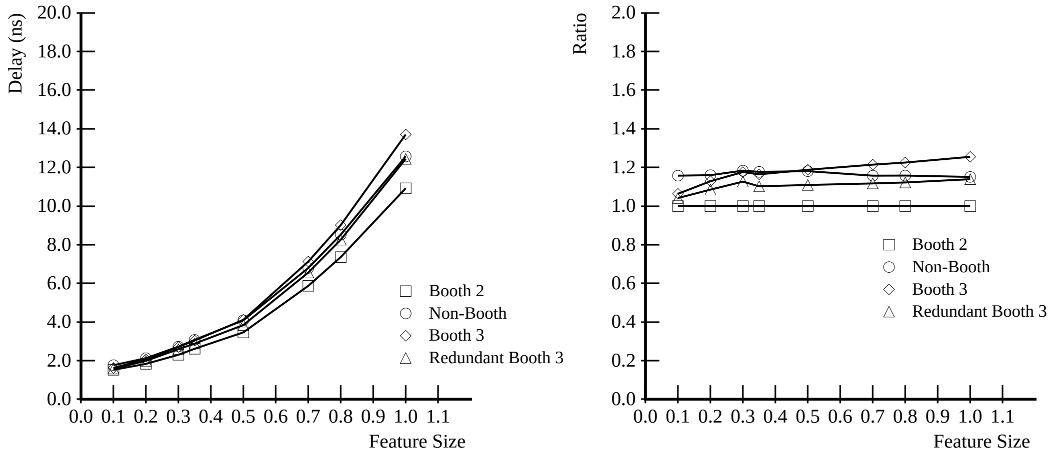


Figure 17: a) Delays (b)Ratio

Figure 17 compares the different topologies if the wires would magically disappear. This is the lower limit for when the wire resistance and capacitance are an insignificant part of the totals for the circuit. The performance advantage for Booth 2 remains unchanged. The ratios between the different Booth algorithms remains fairly constant, as the feature sizes decrease. This is because the capacitances for all the topologies decrease by the same amount. Therefore the total values remain fairly constant as expected.

Booth 2 always provides the smallest latency, while Booth 3 always gives the largest latency. Booth 3 gives the largest latency because, for binary trees it provides no reduction in the number of “4-2” counter levels compared to Booth 2 while still requiring the extra logic to provide the three times multiple. From this figure we notice that non-Booth always gives the second best latency. This occurs because these curves ignore the wire delay, which is significant for the non-Booth, and these curves are related only to the number of transistors in the critical path and the transistor’s loading. Also for this reason, redundant Booth 3 is better than Booth 3 which is the worst.

If there is no wire capacitance, then the curves for the figures will very closely resemble what would happen, if there were no wires. This means that the major effect for the wires is in their capacitance.



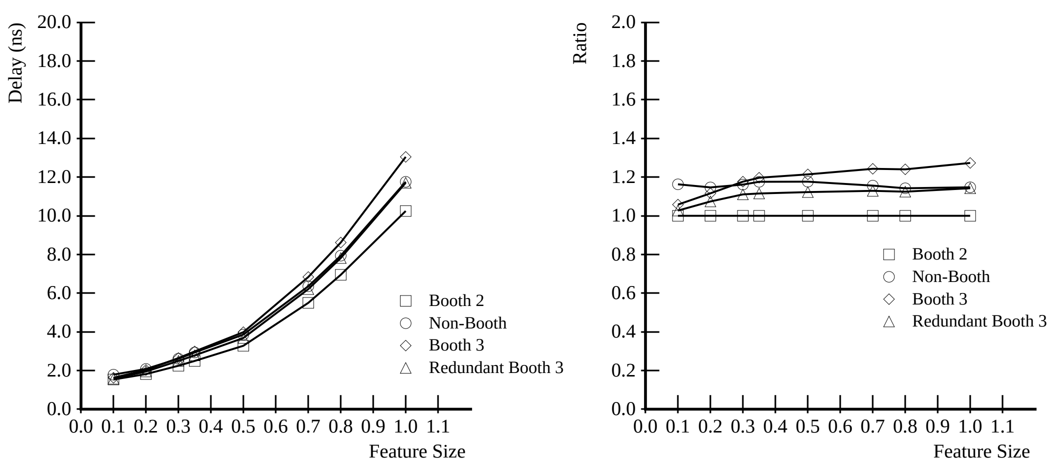
Delay for 4-2 Tree (Slow Wires)

Figure 18: (a) Delays (b)Ratio

If the predicted scaling for the wires does not occur but rather the wires are constantly 25% slower than expected, for all feature sizes, as shown in figure 18 then Booth 2 provides the best performance. However, at small feature sizes redundant Booth 3 provides almost the same latency, but for a smaller area.

In addition, as small feature size decrease, the latency gap between Booth 2 and non-Booth continues to increase. This is because of the long wires in non-Booth. These wires cause non-Booth's capacitance to increase at a larger rate when compared to Booth 2.

However, if the resistance and capacitance of the wires are doubled for each feature size, then at small feature sizes both Booth 3 and redundant Booth 3 provide the same performance as Booth 2.



Delay for 4-2 Tree (Scalable Wires)

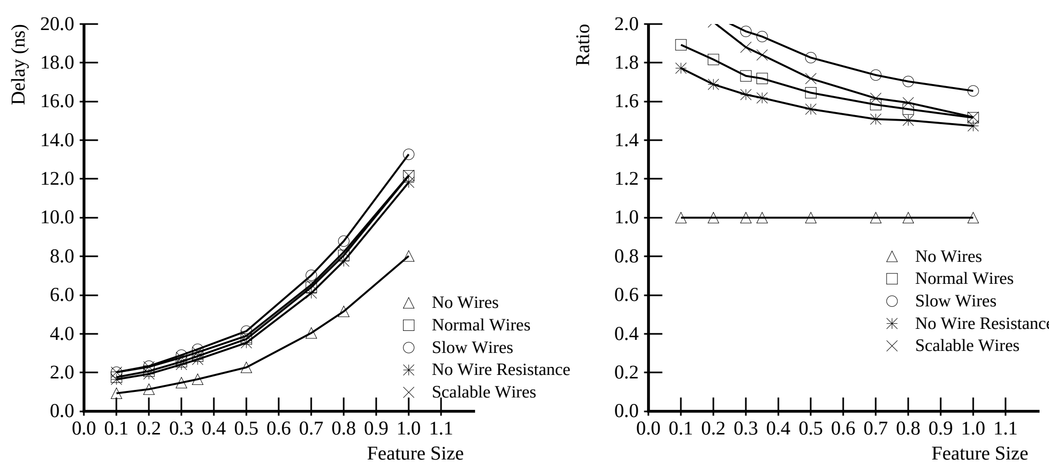
Figure 19: (a) Delays (b)Ratio

However, if wires scale at a slower rate than predicted, as shown in figure 19, then at small feature sizes, Booth 2 does not have an advantage in latency. Booth 3 as expected, shows the biggest improvement. This is because it has the shortest wires so its capacitance increase which is due to wires not scaling as predicted is the least. Redundant Booth 3 also shows the same improvements as Booth 3, though not as dramatically. It provides a multiplier that falls in between Booth 2 and Booth 3 in terms of both area and latency. The improvements of both Booth 3 and redundant Booth 3 leads one to conclude that, if the wires scale at a worst rate than predicted in the future, then Booth 3 and redundant Booth 3 will provide the best latency.

Non-Booth shows a slight deterioration in the ratio of its delay to that of Booth 2 due to its long wires.

4.2 Linear Array

4.2.1 Wire Effects



Booth 2 : Array Wire Differences

Figure 20: Booth 2 : (a) Delays (b)Ratio

The wires have similar effects as binary trees. In the double linear array the wires contribute almost the same percentage of the total delay as they do for binary trees, as shown in figure 20. This occurs although the counters are larger and therefore the total wire length is longer and consequently the wire capacitance is larger. However the ratio of wire capacitance to total capacitance is virtually unchanged because linear arrays have more levels of logic. All the other encoding schemes exhibit the same general characteristics for the different wiring cases and the wire effects on the total delay

The wires have the same effects on the double linear array as they do for the binary tree. This occurs although the linear array has more levels of logic and shorter constant length wires. However, the effects for wires are less for the linear array. The effects are less because wires contribute a smaller portion of the total delay.

This is the Double Linear array using 4-2 Counters. This circuit has more levels of logic, when compared to the binary tree. Its family of curves closely resemble those of the binary tree.

4.2.2 Scaling Effects

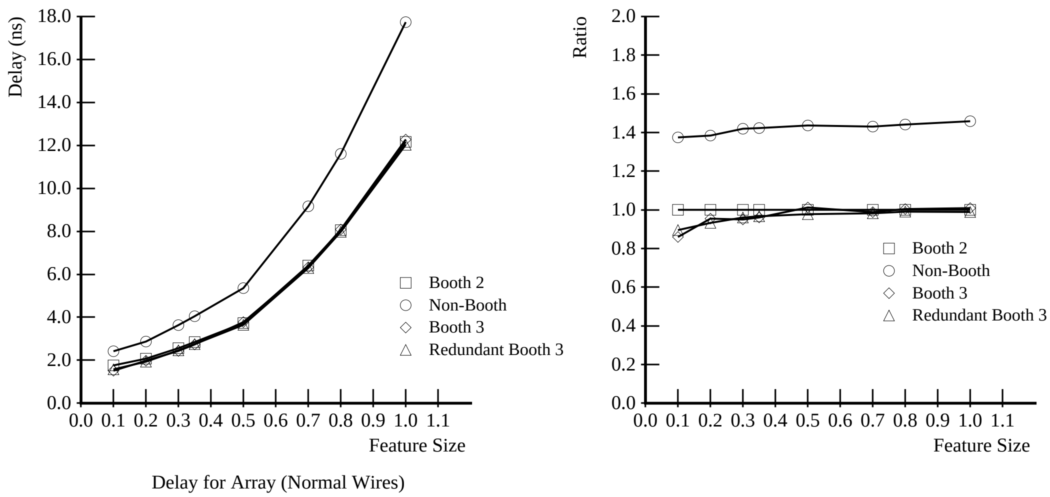


Figure 21: (a) Delays (b)Ratio

Figure 21 gives the performance for the double linear arrays, if the wires scale as expected. Here, Redundant Booth 3 provides the same latency as Booth 2 for large feature sizes and better performance at smaller feature size. Booth 3 provides slightly worse performance at large feature sizes compared to Booth 2, however at small feature sizes it provides the same performance as Redundant booth 3, which is better than Booth 2. The reason why Booth 3 outperforms Booth 2 for this case, is that since these are linear arrays, the number of summands is directly proportional to the number of serial levels of counter in the reduction array. Therefore the extra levels of logic that are needed to provide the three times multiple for Booth 3 are offset by the obtained reduction in the number of summands, and in the shorter wires that are used in the adder. The performance advantage that is achieved by Booth 3 and Redundant Booth 3 is due to the shorter wires that are used in the three times adder. In contrast the ratio of wire capacitance to total capacitance is the same for all these circuits summand reduction arrays.

Non-Booth is consistently slower because of the extra number of counter levels needed to reduce the summands. The curve for the ratio of non-Booth to Booth 2 is almost flat because the array structure has the same wire length between each pair of counters, so the ratio of wire capacitance to total capacitance is the same for both Booth 2 and Non-Booth.

If the wire resistance is negligible then curves resembles what happens when wires are going to scale as expected. However Redundant Booth 3 and Booth 3 are better than Booth 2 for larger feature sizes compared to the normal wires case. This is only because the curves start closer, and not due to any intrinsic differences due to the wires.

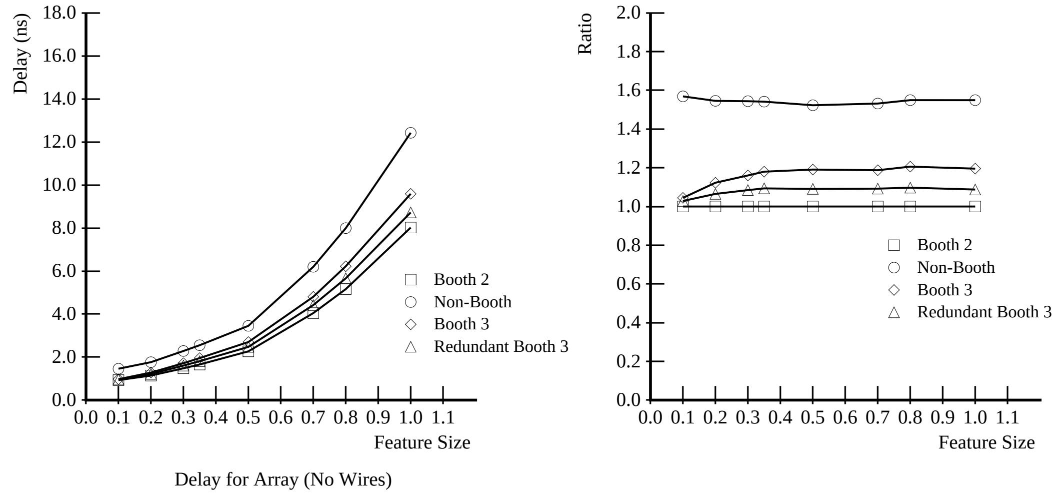


Figure 22: (a) Delays (b)Ratio

The case of there being no wires is represented in figure 22. In this case Booth 2 is the fastest. This is because there are no wires so in effect the difference in latencies is due to the number of transistors in the critical path. Therefore, the topology with the smallest number of transistors (Booth 2) is the fastest. The ratios between the different organizations is relatively constant. The difference in the ratios is due to the different scaling rate for the gate and diffusion capacitances for the transistors.

When there is only wire resistance, in this case these curves resemble the no wire case and the same reasoning applies to this case. This is because the wire resistance is not significant.

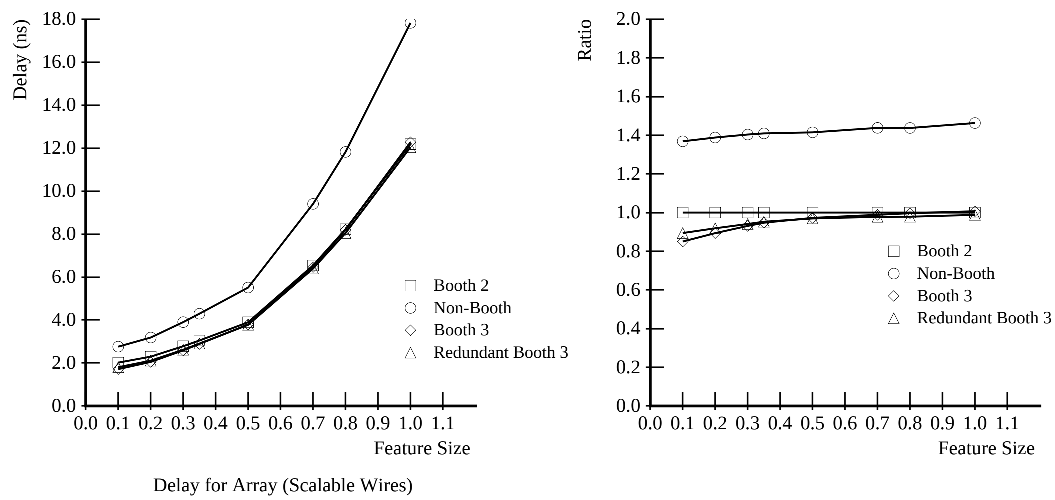


Figure 23: (a) Delays (b)Ratio

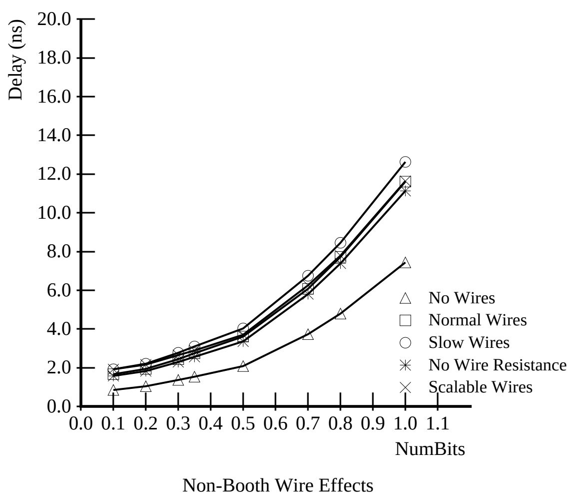
If the wires scale at a slower rate than expected, the latencies are shown in figure 23, then for small feature sizes, both Booth 3 and redundant Booth 3 provide the best possible organizations. It should be noted that redundant Booth 3 is always better than Booth 2. This is because the "4-2" counters are connected serially in the array. It shows an improvement over Booth 2 because it has shorter wires, similarly to that shown by Booth 3.

If our predictions for the wire delays are slightly off and they are slower, then all the curves except for non-Booth are grouped together at large feature sizes. At smaller feature sizes, both Booth 3 and redundant Booth 3 will be faster than Booth 2. This is because of the shorter wires in them.

However, if the predictions for the wires are extremely wrong, then for all feature sizes Redundant Booth 3 gives the smallest latency, with Booth 3 giving comparable latencies for small feature sizes. In this case the performance of the non-Booth also shows some improvement. This improvement is due to the low ratio of wiring capacitance to total capacitance when compared to binary trees.

4.3 Higher Order Array

4.3.1 Wire Effects



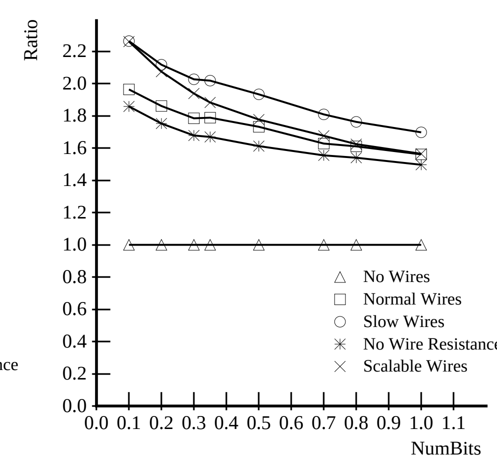


Figure 24: Booth 3: (a) Delays (b)Ratio

The wire contributions for higher order arrays are virtually identical to those of linear arrays, as can be seen from figure 24. This is because double linear arrays are a special case of higher order arrays, and they both use the same counter. In this figure we notice that if wires scale at a worst rate than expected then as feature sizes decrease wires delay contribution approaches that of what would happen if the wire resistance and capacitance was 25% larger than what their values are. The increase in delay exhibited increases linearly with decreasing feature size. This is because the wire capacitance does not decrease as fast expected, so the total capacitance will increase over that of the normal case.

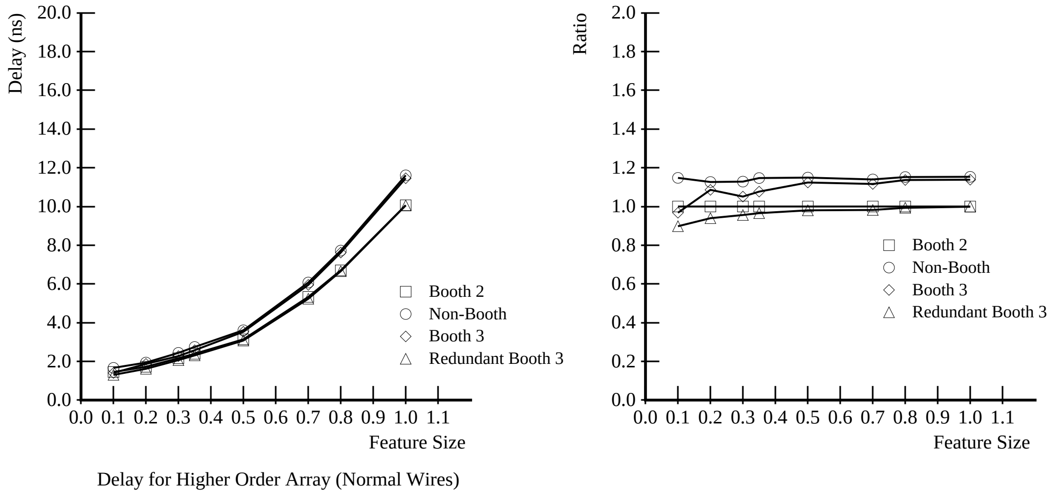


Figure 25: (a) Delays (b)Ratio

4.3.2 Scaling Effects

Figure 25 compares the latencies of the different encoding schemes if the wires scale as expected. For higher order arrays the best latency is provided by redundant Booth 3, its performance advantage increases as the feature size decreases.

Wire resistance is not a factor, and therefore when wire resistance is neglected the latencies are virtually unchanged.

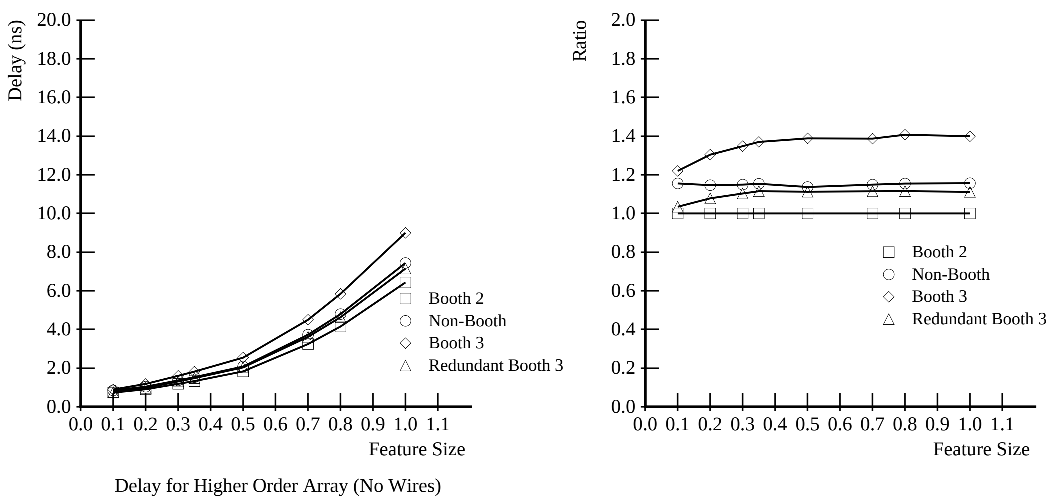


Figure 26: (a) Delays (b)Ratio

Similarly to all the other topologies, when there are no wires Booth 2 provides the best performance as shown in figure 26. This is because Booth 2 has no extra delay needed to generate the 3 times multiple and a simpler selection logic, at the expense of one extra

“4-2” counter level when compared to Booth 3 and Redundant Booth 3. And because it has significantly fewer levels than non-Booth. When the wire capacitance is set to zero and the simulations are performed again, the latencies for the different encoding schemes closely resemble those for the case when there are no wires. Therefore, one can conclude that wire capacitance is the most important parasitic factor for the wires. In the future the domination of the wires capacitance effects is going to increase.

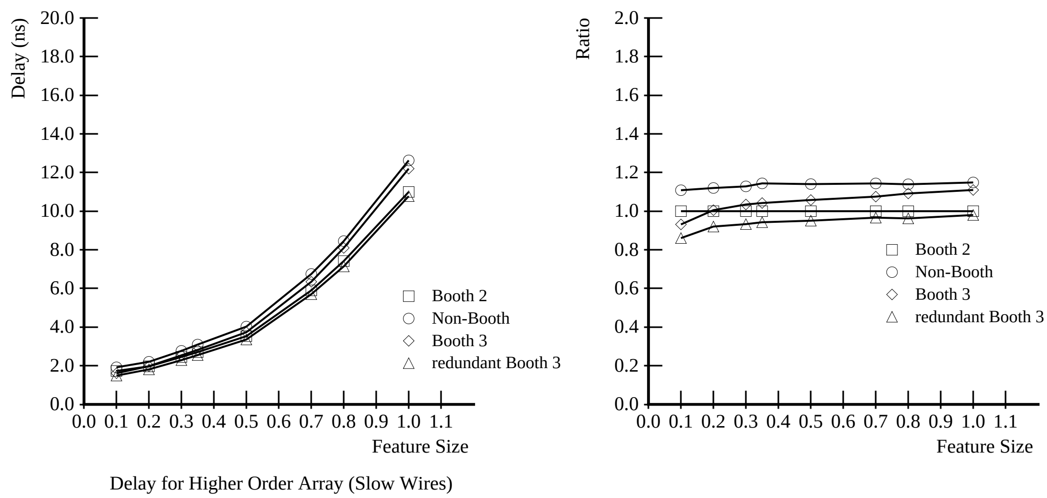


Figure 27: (a) Delays (b)Ratio

For slower wires, redundant Booth 3 provides the best latency. This is because the topology, higher order arrays, has a a square root dependency between the number of summands in the array and the number of counter in the critical path. This means that the number of logic levels is comparable between Booth 2, which has no three times generation but has more counter levels, and redundant Booth 3, that generate the multiple in a redundant form, and therefore needs less levels. Since these are slow wires, their capacitance and resistance values are large. This case favors multiplier organizations that have short wires. As the feature size decreases the wire capacitance value decreases at a slower rate than the active area capacitance. Therefore, at smaller feature sizes the redundant Booth 3 and Booth 3 will have the largest capacitance reduction, due to their shorter wires, and there latency improvement will be the greatest.

If the expectations for the wire capacitance and resistance were overly optimistic, than the advantages for Booth 3 and redundant Booth 3 are magnified.

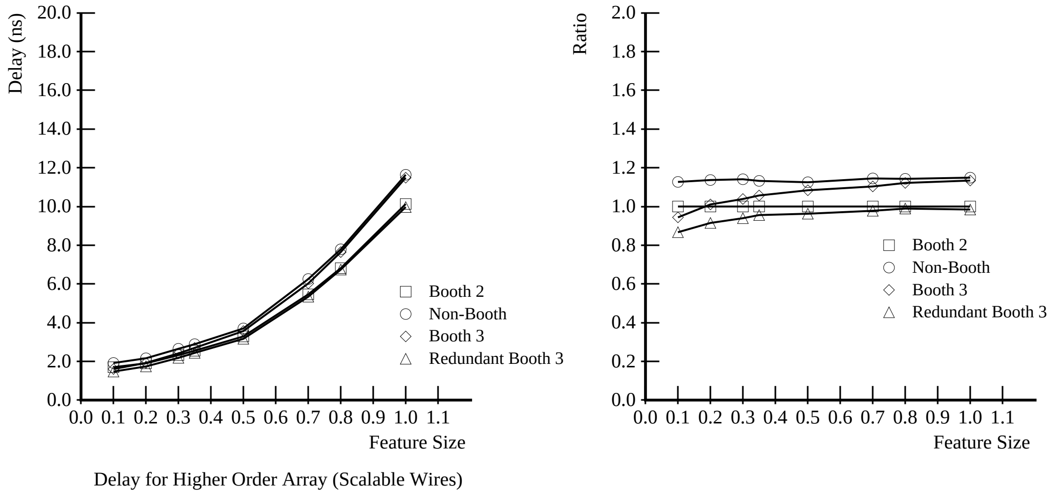


Figure 28: (a) Delays (b)Ratio

If wires scale at a slower rate than expected, then Booth 3 and redundant Booth 3 will provide the best latencies at small feature sizes. That is because at smaller features the wires will be comparable to those of the slow wire case.

4.4 Overturned Staircase Tree

4.4.1 Wire Effects

This is the first topology that makes use of the "3-2" Counter. This has the effect that the number of levels for the tree is increased. This effect is counterbalanced by the fact that overturned-staircase trees give the optimal number of counter levels for almost all number of summands.

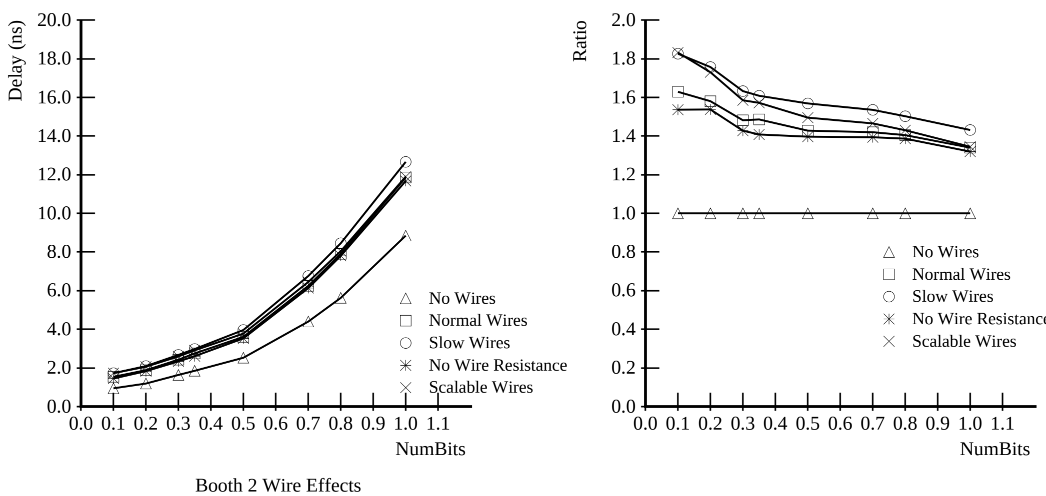


Figure 29: Booth 2 : (a) Delays (b) Ratio

The Effects of the wires' capacitance are greater at smaller feature sizes as can be seen in figure 29. This is due to the wires capacitance providing a greater portion of the total capacitance for the circuit at the smaller feature sizes. The wire capacitance is proportional to both the area and length of the wires, while the capacitance of the transistors is proportional to the areas Therefore, the capacitance of the wires decreases at a slower rate than the capacitance of the transistors i.e. the rate of total capacitance reduction is slower for wires. For extremely small feature sizes the wiring capacitance will become dominated by the fringing capacitance, which is proportional to the wire lengths. Wire capacitance effects are noticeable because it is a significant portion of the total capacitance.

Surprisingly, the wire resistance is not a significant portion of the delay. This is due to the resistance of the wires being a small fraction of the total resistance. In addition as the feature sizes decrease wire resistance values increase slowly, while the resistance of the transistors also increase but at a slower rate. Therefore at smaller feature the wires resistance ratio to the total resistance increases slightly, although the increase in value is so small that its effects on total delay are minimal.

If the wires are going to scale at a slower rate, then their latency at smaller feature sizes is comparable to that of the wires if they are slower by 25%. The increase in the latency is almost linear. The effects of wires are the same for the other three encoding schemes.

4.4.2 Scaling Effects

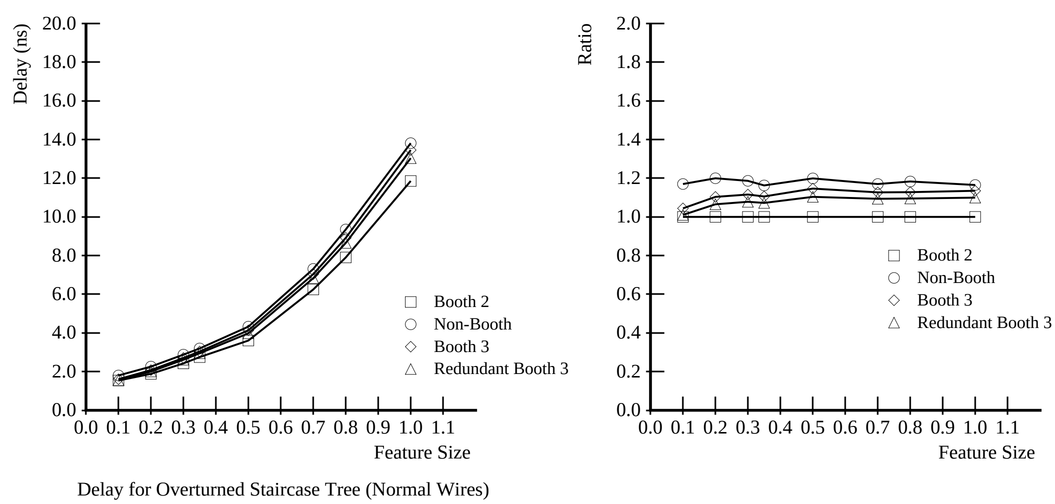


Figure 30: (a) Delays (b)Ratio

If the wires will scale as predicted, then Booth 2 will provide the best latency for Wallace tree, which have the same number of levels as the Overturned Stair-case tree as shown in figure 30. However its latency advantage will decrease at smaller features.

Booth 3 and Redundant Booth 3 show almost the same improvement in this topology because Booth 3 has longer wires in the three times multiple generation while redundant Booth 3 has longer wire in the tree. The difference in wire lengths almost cancels out each other. The increase in the latency of Booth 3 is due to the more complex generation of the multiple.

The case when there is only wire capacitance is very similar. This means that wire resistance is not a significant factor.

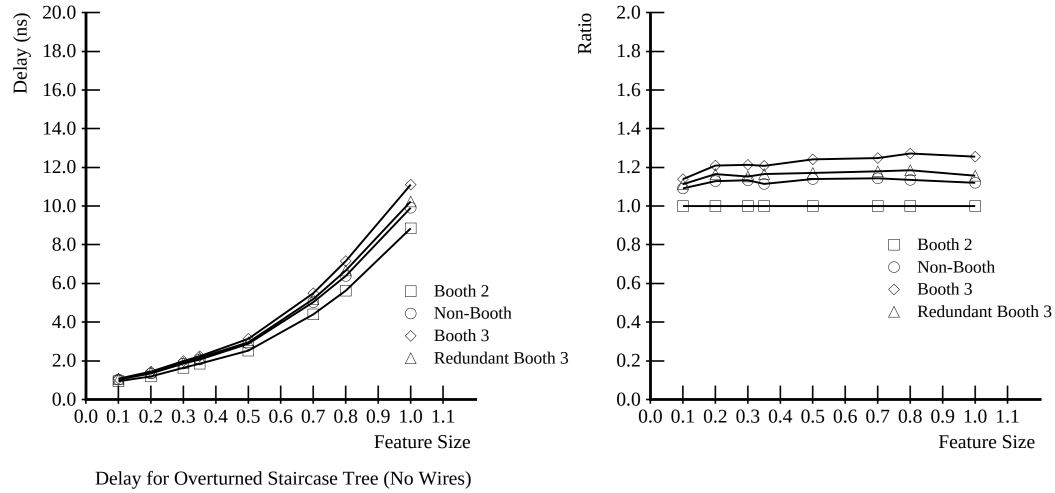


Figure 31: (a) Delays (b)Ratio

When there are no wires the performance advantage for Booth 2 increase. The other topologies will never approach its performance advantage, as can be seen in figure 31. This occurs for similar reasons as the other topologies.

When there is only wire resistance, the curves are very similar to this case. This means that wire capacitance is the dominate effect for wires, and that wire resistance will not become a factor.

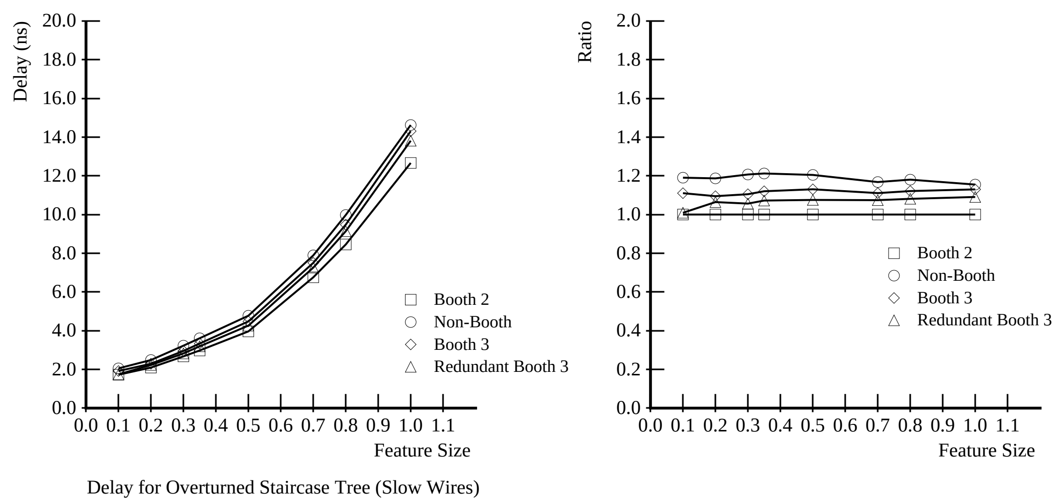


Figure 32: (a) Delays (b)Ratio

For slow wires, Booth 2 remains the best organization, as can be seen in figure 32. This is in contrast to the previous topologies were for slow wires redundant Booth 3 gave the smallest latency. This is because for OS trees both Booth 2 and redundant Booth 3 have the same number of counter levels, and the increase in wire length for Booth 2 over redundant Booth 3 contribution to the total delay is less than that caused by the more complex Booth 3 encoding. However if the wires are extra slow, then at small feature size redundant Booth 3 will provide the best latency. It provides the best latency although it has the same number of counter levels as Booth 2 and it has a more complex summand generation. This is because it has shorter wires, and at the smaller feature sizes when wire capacitance becomes a more significant part of the total capacitance it will have a smaller capacitance total than Booth 2.

If wires scale more slowly than expected. Then at smaller feature sizes the best performance will be provided by Booth 2, Booth 3, and redundant Booth 3. With the higher Booth encoding providing the latency for a smaller area.

4.5 Balanced Delay Tree

This topology uses the "3-2" Counter. This topology has more counter levels than the Overturned Stair-case tree.

4.5.1 Wire Effects

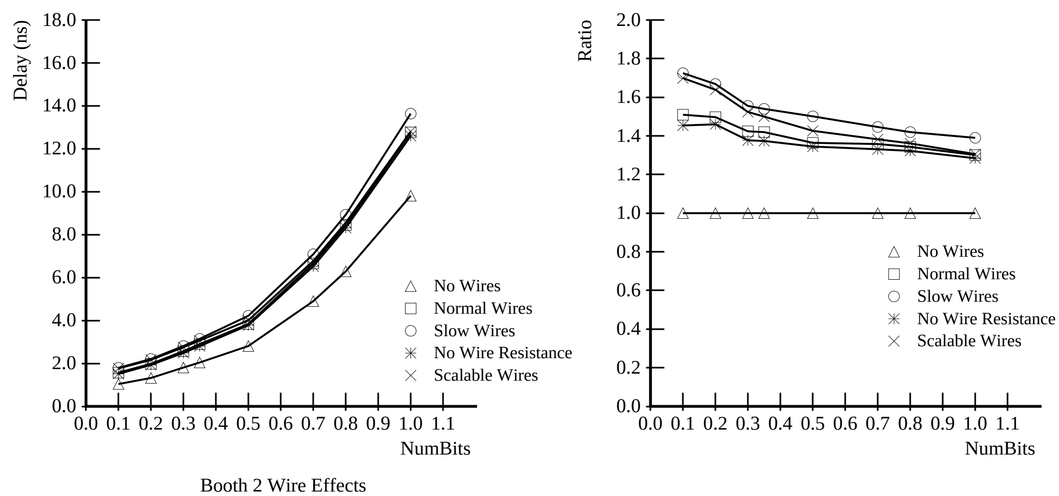


Figure 33: Booth 2: (a) Delays (b) Ratio

The Effects of the wires' capacitance are greater at smaller feature sizes as can bee seen in figure 33. This is due to the wires capacitance providing a greater portion of the total capacitance for the circuit at the smaller feature sizes. The wire capacitance is proportional to both the area and length of the wires, while the capacitance of the transistors is proportional to the areas Therefore, the capacitance of the wires decreases at a slower rate than the capacitance of the transistors i.e. the rate of total capacitance reduction is slower for wires. For extremely small feature sizes the wiring capacitance will become dominated by the fringing capacitance, which is proportional to the wire lengths. Wire capacitance effects are noticeable because it is a significant portion of the total capacitance.

Surprisingly, the wire resistance is not a significant portion of the delay. This is due to the resistance of the wires being a small fraction of the total resistance. In addition as the feature sizes decrease wire resistance values increase slowly, while the resistance of the transistors also increase but at a slower rate. Therefore at smaller feature the wires resistance ratio to the total resistance increases slightly, although the increase in value is so small that its effects on total delay are minimal.

If the wires are going to scale at a slower rate, then their latency at smaller feature sizes is comparable to that of the wires if they are slower by 25%. The increase in the latency is almost linear.

The other encoding schemes also exhibit the same characteristics. This is not surprising since the average wire length for each counter level is the same for all the encoding schemes.

4.5.2 Scaling Effects

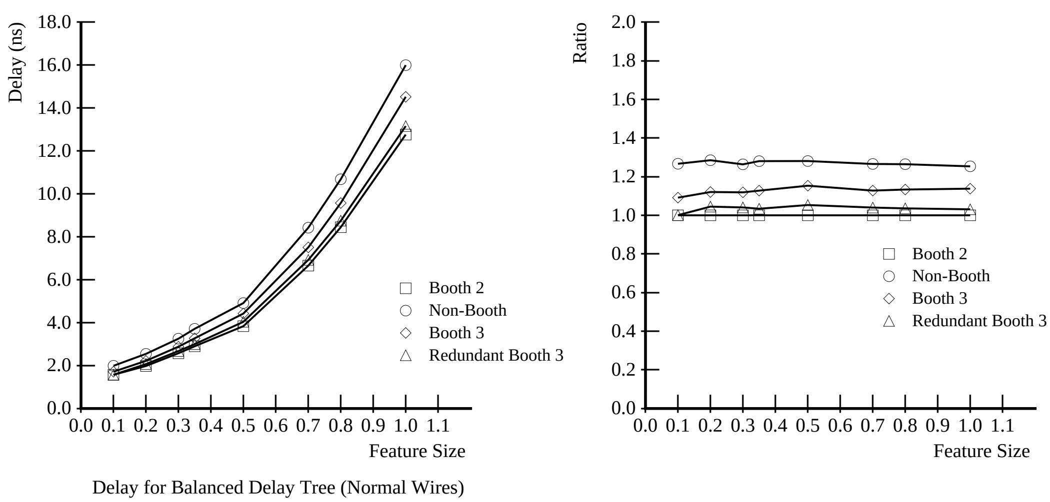


Figure 34: (a) Delays (b)Ratio

For balanced delay trees the best performance is provided by Booth 2. Its advantage, however is not significant over redundant Booth 3 at larger feature sizes and almost non-existent at the smaller feature sizes. Non-Booth similarly to all the other topologies provides the worst latency. Wire resistance and capacitance exhibit the same characteristics as for all the other topologies.

The scaling effects for ZM trees are virtually identical to those exhibited by the OS trees. The only difference is that OS trees have smaller delays.

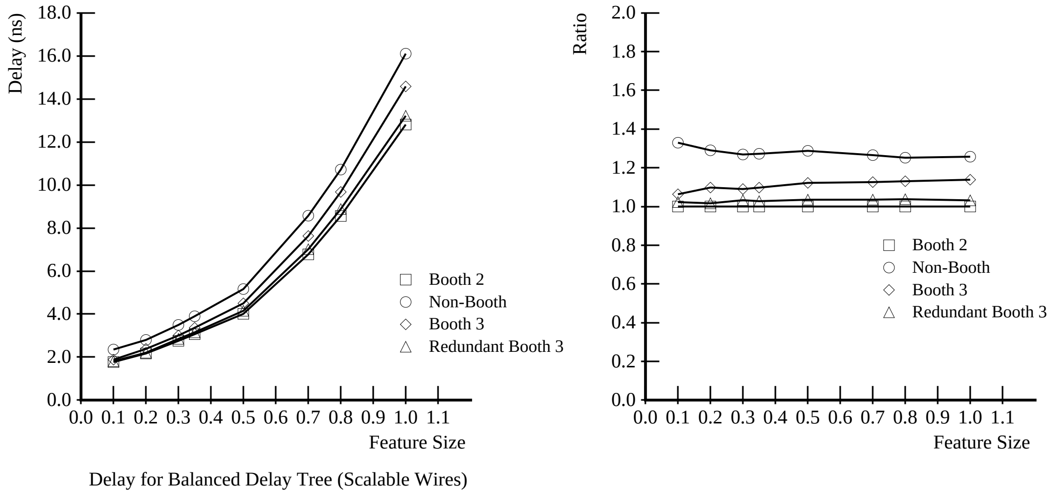


Figure 35: (a) Delays (b)Ratio

If wires scale at a slower rate than expected as shown in figure 35, then Booth 2 and redundant Booth 3 will have virtually the same latency. this also occurs if the wires are slower than expected.

Circuit	Type	Length (λ)	Width (λ)
3-2 Counter	DPL (Single)	255	100
	DPL (Dual)	352	100
4-2 Counter	DPL (Single)	385	100
	DPL (Dual)	493	100
AND 2 Gate	DPL	65	100
	Static	52	100
Booth 2 Encoder	DPL	151	800
Booth 3 Encoder	DPL	238	800
Booth 2 Mux	DPL	151	100
Booth 3 Mux	DPL	238	100
5 Bit Adder	DPL	401	100
56 bit Adder	DPL & Static	804	100

Table 2: Subcell Circuit Sizes

4.6 Areas

The areas for each topology and encoding scheme are expressed as a function of the minimum feature size λ. This is because the design is assumed to go through direct feature size reduction without any major redesign.

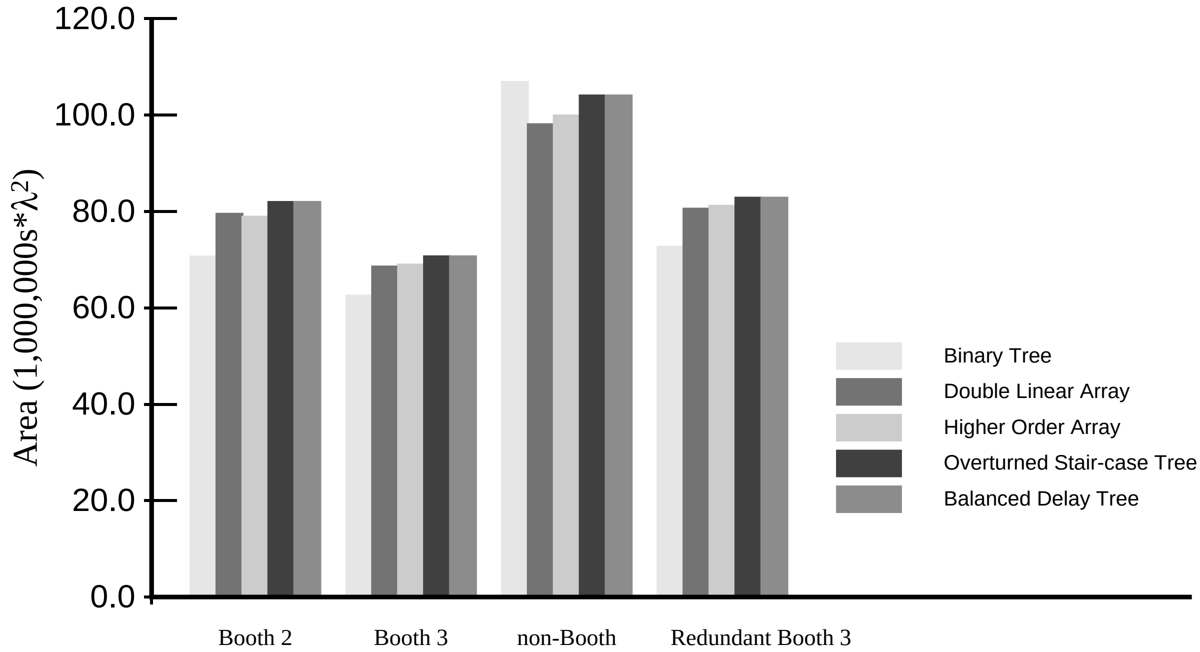


Figure 36: Area for each configuration

The areas for each subcell used in the design are given in table 2. Non-Booth always has the largest area, as can be seen from figure 36. This is because of the large number of summands required. Booth 3 is always the smallest because it requires the fewest summands. However, the area for Booth 2 is slightly smaller than or equal to the area for redundant Booth 3. This is because of the need for an adder in redundant Booth 3 and that the Booth muxes are larger for Booth 3, which offsets the savings in area due to the reduction in the number of counters.

5 Conclusion

Non-Booth is not a viable design. It consistently gives the largest area, and always is one of the worst in terms of latency. Booth 2 gives the designs with the smallest latency. This is because an additional adder is not required, and that the best possible reduction in number of counter levels needed to sum the summands, as achieved by Booth 3, is only 1. Booth 3 produces the smallest designs because they have the least number of summands. Redundant Booth 3 is not very attractive for tree based designs. It is more suited to standard cell based designs, in which higher order arrays can be thought of as an extreme case, because the number of levels is more closely correlated with the number of summands.

As wires continue to account for larger fractions of the total delay, due to decreasing feature size, redundant Booth 3 provides the best best latency. This occurs because of the shorter wires that are needed by Booth 3 and redundant Booth 3. Without any wires Booth 2 always gives the smallest latency irrespective of the topology chose.

There are no area savings inbuilding redundant Booth 3 compared to Booth 2. The savings in area due to reduction in the number of counters is offset by the increase in area that occurs because of the need to generate the multiple and because of the larger decoders.

The binary tree is the most attractive solution in terms of both area and latency. It has the smallest latency and area of all the different topologies for all feature sizes. This occurs because of the new "4-2" counter design which has 3 serial XOR gates instead of the 4 serial that is obtained by the simple contacting of two "3-2" counters. There is also an area saving when compared to two "3-2" counters.

References

[1] An American National Standard, "IEEE Standard for Floating Point Arithmetic", *ANSI/IEEE standard 754-1985*

[2] O.L. McSorley, "High Speed Arithmetic in Binary Computers", *Proceedings of the IRE*, 49(1), pp. 67-91, Jan. 1961.

[3] S. Vassiliadis, E. Schwarz and B. Sung, "Hard-wired Multipliers with Encoded Partial Products." *IEEE Trans. on Computers*, vol 40, No.11, pp. 1181-1197, Nov. 1991.

[4] G. Bewick, "Binary Multiplication Using Partially Redundant Multiples",*Technical Report: CSL-TR-92-528*, Stanford University, June 1992.

[5] M. Santoro, "Design and Clocking of VLSI Multipliers", *Ph.D. Thesis, Stanford University*, Oct. 1989.

[6] D. Zuras and W. McAllister, "Balanced Delay Trees and Combinatorial Division in VLSI,"*IEEE J. Solid-State Circuits*, vol SC-21, No.5, pp. 814-819, Oct. 1986.

[7] C. S. Wallace, "A Suggestion for a Fast Multiplier", *IEEE Trans. Electronic Computers*, pp. 14-17, Feb. 1964.

[8] Z. Mou and F. Jutand, "A Class of Close to Optimum Adder Trees allowing Regular and Compact Layout", *IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors*, pp.251-254, 1990.

[9] G. McFarland and M. Flynn, "Limits of Scaling MOSFETs", *Technical Report: CSL-TR-95-662 Revised*, Stanford University, Nov. 1995.

[10] N. Ohkubo et al., "A 4.4 ns CMOS 54*54-b Multiplier using Pass-Transistor Multiplexor", *IEEE J. Solid-State Circuits*, vol SC-30, No.3, pp. 251-257, Mar. 1995.