A Composite Arithmetic Scheme for Evaluation of Multinomials*

Pavan Adharapurapu
Computer Science Department
Univ. of California at Los Angeles
pavan@cs.ucla.edu

Miloš D. Ercegovac
Computer Science Department
Univ. of California at Los Angeles
milos@cs.ucla.edu

Abstract: **We discuss the implementation aspects of the Multinomial Online Evaluation (MOLE) scheme [1] and compare its delay and cost with three other schemes. The MOLE scheme is meant for the evaluation of a generic multinomial represented as an evaluation graph. In this paper, we compare the MOLE scheme with the following schemes: A network of online adders and multipliers, a network of online adders and MLSOs, and a network of conventional multioperand adders and multipliers. A strawman example in the form of a three-variable multinomial and its evaluation graph is used to do the comparison analysis. Compared to the conventional implementation, the MOLE scheme is estimated to have favorable delay at a higher cost.**

1 Introduction

Consider the following three-variable multinomial:

$$\mathcal{M}(x,y,z) = x + y + \frac{1}{12}x^2y + \frac{1}{23}xyz^2 + \frac{1}{23}xy^3z^3 \quad (1)$$

An E-graph of this multinomial is shown in Figure 1. An evaluation graph (E-graph) is a rooted directed acyclic graph (DAG) representation of a multinomial in a factored form. It is important to note that an E-graph need not necessarily have a tree structure.

The goal of this paper is to assess the recently proposed [1] Multinomial On-Line Evaluation (MOLE) scheme for evaluating multinomials. To do this, we discuss the implementation of the MOLE method and compare its delay and cost against three other schemes for evaluation

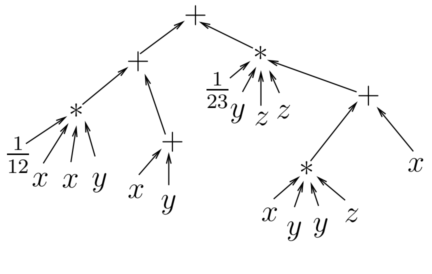


Figure 1: An E-graph corresponding to the multinomial $\mathcal{M}$

of the above E-graph. We hope, through this exercise, to bring out the advantage of the MOLE method for evaluating arbitrary multinomials.

The current research is prompted by recent discoveries [2] in the Bayesian Networks field relating to the representation of a Bayesian Network using a characteristic multinomial. In addition to this, algorithms have been proposed which represent the Bayesian Network Multinomial (which is exponential in size) as a compact E-graph. Evaluation of this E-graph is required for *probabilistic inference*, the most important operation performed on Bayesian Networks. Even after representing them as compact E-graphs, such multinomials are not amenable to software evaluation for real-time applications.

In the rest of the paper, we compute the delay and cost of the MOLE scheme and three other schemes viz., Scheme I (Network of Online Adders and Multipliers), Scheme II (Network of Online Adders and MLSOs[1]) and Scheme III (Network of Conventional Multioperand Adders and Multipliers). We summarize the results in the final section.

In calculating delay and area estimates for the various schemes, we will use the following table of values for standard components. We use the area/delay of a Full Adder (a_{FA} and t_{FA}) as the unit area/delay. We do not consider estimates for wiring, layout, etc.

*Copyright 2004 IEEE. Published in the Proceedings of the 38th Annual Conference on Signals, Systems, and Computers, November 7-10, 2004, Asilomar, Pacific Grove, CA. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works, must be obtained from the IEEE.

[1]introduced later

TABLE 1: DELAY/COST VALUES PER BIT FOR STANDARD COMPONENTS

	Area	Delay
FA	1.0	1.0
[4:2] cell	2.0	1.5
[5:2] cell	2.5	2.0
2:1 mux	0.5	0.5
Flip Flop	0.6	1.0
Buffer	0.5	0.5
Digit selection	3.0	1.5

2 The MOLE scheme

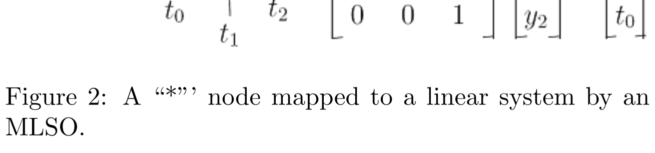
The MOLE scheme consists of the following steps:

- Applying Linear-System Operators (LSOs) to convert sub-graphs of the E-graph into systems of linear equations.
- Solving these linear systems using the E-method [3] in an online fashion.

An LSO maps an E-graph (or a sub-graph) into an equivalent system of linear equations. Equivalence means one of the unknowns (typically y_0) of the linear system has the same value as the graph. Two types of LSOs are defined - the Multiplication LSO (MLSO) and the Polynomial LSO (PLSO). The former maps a multiplication ("*") node into a linear system while the latter does the same for an alternating string of addition ("+") and multiplication nodes satisfying the following constraints: the "+" nodes have only two children, all nodes have only one parent and the lowest node of this string is a "*" node. MLSO and PLSO are illustrated in Figures 2 and 3 respectively.

Application of the LSOs is done using the following algorithm:

1. Arrange the E-graph into levels based on maximum distance from root node.
2. Do a breadth first search starting from the root to identify all maximal sub-graphs that can be mapped using a PLSO and replace them by a PLSO.



$p = t_0t_1t_2$
*
t_0
t_1
t_2
$\begin{bmatrix} 1 & -t_2 & 0 \\ 0 & 1 & -t_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ t_0 \end{bmatrix}$

Figure 2: A "*"' node mapped to a linear system by an MLSO.

3. Map the rest using MLSOs and online adders(OLAs).

It can be shown [1] that it is more efficient, delay wise, to map a PLSO-mappable string-of-nodes using a PLSO rather than a combination of MLSOs and OLAs. Hence, the emphasis on PLSOs in the above algorithm.

The result of applying the MOLE scheme is a network of LSOs which are serially connected. The network is timed so that all LSOs on a given level start simultaneously. This starting time is determined by when the online module with the largest online delay in the immediately lower level has produced its first output bit. This arrangement may require buffers to hold output bits from modules with unequal online delays (although this is not required for the example E-graph).

We use an online method - the E-method [3] - for solving system of linear equations. For a system with n non-trivial unknowns, the E-method uses n online multiply-add modules (connected serially), each computing one of the unknowns in a digit-by-digit manner starting with the most significant digit. "Trivial" unknowns given by equations such as $y_i = t$ don't need an online module. They are either produced by another module or, in the case when it is a direct input, can be extracted serially from the input register. The digit set for the output is {-1,0,1}.

All the coefficients of the linear system are in a serial form. The i^{th} module executes the following recurrence equation for as many iterations as the required precision of the output:

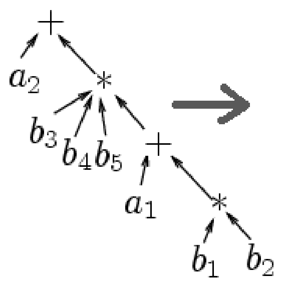
$$\begin{aligned} w(i)[j] \;=\;& 2(w(i)[j-1] + 2^{-(\delta+1)}b(i)_{(\delta+j)} - \\ & d(i)[j-1] + a(i)[j-1]\; d(i+1)[j-1] + \\ & 2^{-(\delta+1)}\; a(i)_{(\delta+j)}\; D(i+1)[j-1]) \end{aligned} \quad (2)$$

The notation used is as follows: the number in the square brackets indicates iteration number, that in braces indicate the row number and the subscript indicates the fractional digit's position. Convergence of the above recurrence is satisfied if all inputs are less than 1/8. The online delay, δ, is 2. Thus, δ of MLSO and PLSO is 2 cycles irrespective of the number of operands.

For further details on the MOLE method, please consult [1].

2.1 Delay and Cost

In this subsection, we calculate the area and cost of the MOLE scheme for the E-graph of Figure 1. We shall assume that the inputs for the multinomial are sufficiently small so that the convergence requirements of the algorithm driving the MOLE scheme are satisfied without requiring the pre-scaling of inputs. While this may not hold true for all applications, it does hold for many classes of



$$\xrightarrow{\quad} a_2 + b_3 b_4 b_5 (a_1 + b_1 b_2) \xrightarrow{\quad} \begin{bmatrix} 1 & -b_3 & 0 & 0 & 0 \\ 0 & 1 & -b_4 & 0 & 0 \\ 0 & 0 & 1 & -b_5 & 0 \\ 0 & 0 & 0 & 1 & -b_1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} a_2 \\ 0 \\ 0 \\ a_1 \\ b_2 \end{bmatrix}$$

Figure 3: A string of nodes mapped to a linear system by a PLSO.

applications such as Bayesian Networks (which deal with probabilities).

Figure 4 shows the resulting circuit obtained by applying the MOLE method. As mentioned earlier, we first do a top-down search to find possible subgraphs for PLSO application. These are shown shaded in grey. We map the rest using MLSOs and OLAs. The example E-graph requires two PLSOs, and one OLA. It does not require any MLSOs.

The online module is implemented using a [5:2] adder (delay, [4:2] for area[2]), four registers, two muxes with complementers and a digit SEL unit. See [4, pg. 576] for more details on the module implementation. Thus, the area of each m-precision module is given by:

$$\begin{aligned} A_{module} &= a_{[4:2]} + 4 \cdot a_{REG} + 2 \cdot a_{MUX} + a_{SEL} & (3) \\ &= (5.4m + 3.0)\ a_{FA} & (4) \end{aligned}$$

Note that only the first three terms of (3) depend on m. The cycle time for the module is given by:

$$\begin{aligned} T_{cycle} &= t_{[5:2]} + t_{REG} + t_{MUX} + t_{SEL} \\ &= 5\ t_{FA} & (5) \end{aligned}$$

In Figure 4, the "root" PLSO has seven non-trivial unknowns and hence requires seven online modules of the kind mentioned before. The other PLSO has three non-trivial unknowns and requires three online modules. The remaining OLA is a simple two-operand online adder whose area will later be estimated to be 5 $a_F A$.

Thus, the total area for the MOLE implementation for an output precision of m is given by:

$$\begin{aligned} A(MOLE) &= 7 \cdot (5.4m + 3.0) + 3 \cdot (5.4m + 3.0) + 5 \\ &= (54m + 35)\ a_{FA} & (6) \end{aligned}$$

The delay analysis is simple in this case. Any LSO has an online delay of 2 cycles. This follows directly from the recurrence (2). By definition of online delay, the first output bit is obtained after $2 + 1 = 3$ cycles. The two-operand online adder has an online delay of, again, 2 cycles. Thus, the first bit of the final output (the multinomial value) is obtained after 9 cycles and all m bits of the output are obtained after

$$T(MOLE) = (9 + m)\ t_{cycle} = (45 + 5m)\ t_{FA} \qquad (7)$$

Figure 4 lists the component-wise breakup of areas and delays.

We can also calculate the area and delay when we only have access to modules of precision m/k. The E-method allows a linear tradeoff between latency and area. With k times smaller-precision modules, one has to perform each iteration k times to cover the full precision. The total area and delay, consequently, are $(54m/k + 35)\ a_{FA}$ and $(45 + 5km)\ t_{FA}$ respectively.

3 Scheme I

This scheme involves replacing the "*" and "+" nodes of the E-graph with multioperand online adders and multipliers. Each multioperand adder and multiplier will be built as a tree network of two-operand online adders and multipliers. This is equivalent to expanding the graph so that all nodes have exactly two children and replacing each node with a two-operand online adder/multiplier. As in the MOLE scheme, all units (adders/multipliers) in a given level start execution simultaneously and may require output buffers.

3.1 Delay and Cost

As mentioned above, each "*" node is replaced by a tree of two-operand online multipliers. The δ of two-operand online multiplier is 3 cycles. The cycle time for a two-operand online multiplier is roughly the same as the E-method online module because of similar structure [4,

[2]this is because most of the unknowns of the PLSO linear system have the RHS matrix coefficient as zero and can be implemented using just a [4:2] adder.

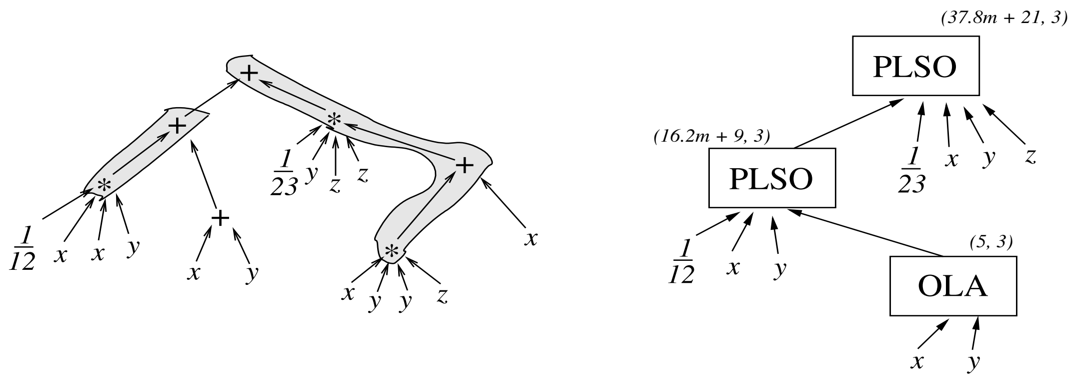


Figure 4: Circuit obtained after application of MOLE scheme. The pair (a,d) on the LSOs denotes (area, online delay).

pg.520]. Thus, an n-operand online multiplier has an online delay of:

$$\begin{aligned} \delta_{n-mult} &= (3+1)\cdot log(n)\cdot t_{cycle} \\ &= 4\cdot log(n)\cdot t_{cycle} \\ &= 20\cdot log(n)\ t_{FA} \qquad (8) \end{aligned}$$

A two-operand online adder has a δ of 2 cycles. The cycle time for this adder is less than that of the two-operand multiplier. Because of the need for a common clock, the cycle time is fixed to the larger of the two. Thus, an n-operand online adder has an online delay of:

$$\begin{aligned} \delta_{n-add} &= 3\cdot log(n)\cdot t_{cycle} \\ &= 15\cdot log(n)\ t_{FA} \qquad (9) \end{aligned}$$

The example E-graph has five levels when expanded. Except for the root level, every other level has a "*" node which means the online delay for these levels will be 3 cycles each. The root level has a δ of 2. Thus, the total delay for m bits of the output is:

$$\begin{aligned} T(SchemeI) &= (4\cdot 4+3+m)\ t_{cycle} \\ &= (95+5m)\ t_{FA} \qquad (10) \end{aligned}$$

A two-operand online multiplier has a similar structure to the online module unit and its area, thus, is $(5.4m + 3.0)\ a_{FA}$.The two-operand online adder is built using two FAs and five latches (see [4, pg. 506]). Consequently, its area is 5.0 t_{FA}. Applying Scheme I for the example E-graph requires ten such multipliers and four such adders. Thus, the total area for Scheme I is:

$$\begin{aligned} A(SchemeI) &= 10\cdot(5.4m+3.0)+4\cdot 5.0 \\ &= (54m+50)\ a_{FA} \qquad (11) \end{aligned}$$

4 Scheme II

Scheme II differs from Scheme I only in the fact that the "*"' nodes of the E-graph are replaced by MLSOs instead of online multipliers. This is clearly advantageous in terms of delay since the online delay of an MLSO is 2 cycles irrespective of the number of operands.

4.1 Delay and Cost

When Scheme II is applied to the example E-graph, there will only be MLSOs and online adders in the resulting circuit, all of which have a δ of 2 cycles. Since there are four levels in the expanded E-graph, the total online delay of the circuit is 12 cycles and the total delay for the Scheme II to produce m bits of the output is:

$$T(SchemeII) = (12+m)\ t_{cycle} = (60+5m)\ t_{FA} \qquad (12)$$

Scheme II requires four OLAs and three MLSOs. All three MLSOs require ten online modules in total. Thus, the total area for Scheme II is:

$$\begin{aligned} A(SchemeII) &= 4\cdot 5.0+10\cdot(5.4m+3.0) \\ &= (54m+50)\ a_{FA} \qquad (13) \end{aligned}$$

5 Scheme III

In this scheme, each of the "*" and "+" nodes are replaced by conventional arithmetic units to obtain a network of conventional arithmetic modules. Each "*" node is replaced by a tree of two-operand multipliers and a register. Each "+" node is replaced by a tree of two-operand adders and a register. Again, this is the same as expanding the E-graph so that each node has exactly two children and replacing each node with the two-operand

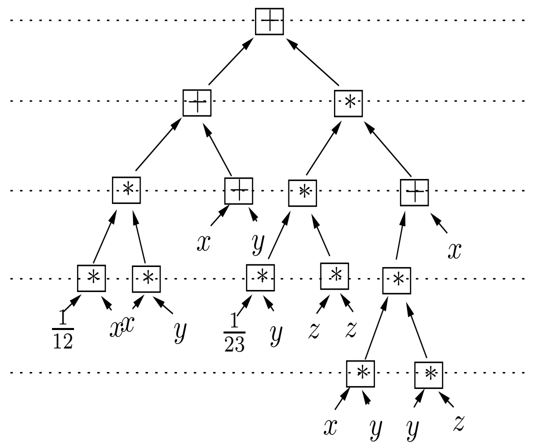


Figure 5: Circuit obtained after application of Scheme III.

adders/multipliers. This expanded graph is arranged in levels and the levels are timed as described previously so that all nodes on a given level start simultaneously. Figure 5 shows the final circuit obtained after applying Scheme III.

5.1 Delay and Cost

For the two-operand multiplier we use a radix-2 serial-parallel multiplier and for the adder we use a carry-ripple adder. A similar analysis is applicable to a radix-4 serial-parallel multiplier using a Booth recoder. The multiplier has an on-line delay of m cycles since only the most significant half of the product is used. The total delay is

$$T_{SPmult} = (t_{mux} + t_{FA} + t_{reg})m = 2.5m\ t_{FA}$$

The additions are overlapped with the generation of the MS half of the product with an on-line delay of one cycle. A serial-multiplier has a cost

$$\begin{aligned} A_{SPmult} &= a_{mux} + a_{[3:2]} + 2a_{reg} \\ &= (0.5 + 1 + 1.2)m \\ &= 2.7m\ a_{FA} \end{aligned} \tag{14}$$

The cost of the adder is $2m\ a_{FA}$ including the sum register.

For the example E-graph, we would need ten two-operand multipliers and four two-operand adders. Hence the total area for Scheme III is:

$$\begin{aligned} A(SchemeIII) &= (10 \cdot 2.7m + 4 \cdot 2m)\ a_{FA} \\ &= 35m\ a_{FA} \end{aligned} \tag{15}$$

The expanded E-graph has five levels. Its delay is:

$$T(SchemeIII) = 5m\ cycles = 12.5m\ t_{FA} \tag{16}$$

Note that for this scheme($r = 2$), 1 $cycle = 2.5\ t_{FA}$. The values for $r = 4$ are obtained in a similar way.

6 Summary

The following table summarizes the delay and area requirements for all the four schemes for $m = 32$. The MOLE

TABLE 2: DELAY/COST COMPARISIONS FOR THE VARIOUS SCHEMES

	Area	Delay
MOLE	1763	205
Scheme I	1778	255
Scheme II	1778	220
Scheme III (r=2)	1120	400
Scheme III (r=4)	1440	280

scheme has lowest delay of all the schemes discussed.

To summarize, the MOLE scheme is a hardware-oriented method to evaluate multinomials. Its advantage is that it can adapt to any E-graph of the multinomial. MOLE method is online and the output is available after a few cycles. This can be used to chain it with other operations. MOLE method has favorable cost and delay properties. The implementation uses simple modules, whose cycle time is independent of precision. The modules are connected digit-serially with minimal interconnect. Simple area/time tradeoffs are also possible. A conventional scheme (r=4) has a 18% smaller cost but a 36% longer delay.

References

[1] P. Adharapurapu and M.D. Ercegovac, "A Linear-System Operator based Scheme for Evaluation of Multinomials", *Internal Report, Computer Science Department, UCLA*, June 2005.

[2] A. Darwiche, "A Differential Approach to Inference in Bayesian Networks", *Journal of the ACM*, vol. 50, no. 3, pp. 280-305, May 2003.

[3] M.D. Ercegovac, "A General Hardware-Oriented Method for Evaluation of Functions and Computations in a Digital Computer", *IEEE Transactions on Computers*, vol. C-26, no. 7, July 1977, pp. 667-680.

[4] M.D. Ercegovac, T. Lang, *Digital Arithmetic*, Morgan Kaufmann, 2004.