Radix-2^n Serial-Serial Multipliers

Authors: A AGGOUN, A FARWAN, M K IBRAHIM* and A ASHUR

Address: Faculty of Computing Sciences and Engineering, De Montfort University, The Gateway, Leicester LE1 9BH.

*Department of Computer Engineering, P.O.Box 1367,
King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia.

Contact: Dr A Aggoun

Tel: +44 (0) 116 2577055
Email: aggoun@dmu.ac.uk

Abstract

All serial-serial multiplication structures reported in the literature has been confined to bit serial-serial multipliers. In this paper, an architecture for digit serial-serial multipliers is presented. A set of designs are derived from the radix-2^n design procedure which was first reported by the authors for the design of bit level pipelined digit serial-parallel structures [8]. One significant aspect of the new designs is that they can be pipelined to the bit level and give the designer the flexibility to obtain the best trade off between throughput rate and hardware cost by varying the digit size and the number of pipelining levels. Also in this paper an area efficient digit serial-serial multiplier is proposed which provides a 50% reduction in hardware without degrading the speed performance. This is achieved by exploiting the fact that some cells are idle for most of the multiplication operation. In the new design, the computations of these cells are re-mapped to other cells, which make them redundant. The new designs have been implemented on the S40BG256 device from the SPARTAN family to prove functionality and assess performance.

1. Introduction

Bit-serial computations have been studied extensively during the last decade. Complete VLSI design methodologies based on bit-serial arithmetic have been built, particularly for the VLSI implementation of digital signal processing (DSP) algorithms [1, 2]. However, the disadvantage of the bit-serial systems is that they are considered to be very slow for many applications such as radar, sonar, and video and image processing. For higher throughput DSP applications, however, it is clear that a move towards higher bit-width operations is necessary, where the solution often used is bit-parallel computation. However, the bit-parallel systems require a very large amount of silicon area, communication overhead, and pin-out.

To avoid the disadvantages of the bit-serial and the bit-parallel computation, the concept of digit serial-parallel implementations have been proposed in recent years [3-11]. Several approaches have been proposed to design digit serial-parallel architectures based on two's complement number representation. Early on most of the proposed design methodologies use the *bit-level cellular arrays* as their *starting* point [3-5]. The major drawback of the architectures based on these approaches is that they cannot be pipelined at a sub-digit level due to the existence of carry feedback loops. This has severely limited their throughput, which could be a major obstacle for high speed applications. In digit serial-parallel systems, the throughput rate often represents the overriding factor dictating system performance.

Recently, a structured design methodology of digit serial-parallel structures has been proposed by the authors based on the Radix-2^n arithmetic [6-9]. It is based on the linear projection of the Dependency Graphs of the radix-2^n digital signal processing algorithms according to the classical

theoretical framework of regular array architectures synthesis [8]. This has resulted in the design of the first digit serial-parallel structures which can be pipelined to the sub-digit level [6-9].

Other procedures have been proposed in literature to design digit serial-parallel structures [10-11]. These are based on turning the bit serial-parallel multipliers to digit serial-parallel multipliers by replacing the AND gate and the Full Adder (FA) in each cell with an n-bit multiplier and n-bit carry save adders, respectively. Where n is the digit size. Although, these procedures would result in a variety of digit serial-parallel multipliers which can be pipelined to the bit level, they do not cover the whole spectrum of possible architectures. For example, the product of two n-bit numbers is a 2n-bit number which can be divided into most and least significant digits and depending on their transfers from one significance to another, a number of architectures can be derived as shown by the authors in [8]. This is not possible if the starting point in the design process is the bit serial-parallel multiplier. On the other hand using the radix-2^n design procedure reported in [8], all possible architectures can be derived due to the generalised description of the multiplication using the radix-2^n arithmetic. Secondly the radix-2^n approach allows the direct application of the existing synthesis methods and design tools such as DG2VHDL [12] in implementing digit-serial structures.

The possibility of high degree of pipelining has led to an increase in the throughput rate of the digit serial-parallel systems at the cost of a small increase in hardware cost. It was shown that sub-digit pipelined digit serial-parallel architectures designed using the proposed radix-2^n based methodology can achieve a *better* throughput rate at a much lower hardware cost than the bit-parallel structure [8, 11].

The bit level pipelined digit serial-parallel techniques are ideal in Digital Signal Processing as they offer the flexibility, which is needed in order to exploit the potential of the available technology [8]. However, they still require one of the input data to be fed in parallel, which increases the number of input/output ports especially in dense designs. With FPGAs the size becomes another obstacle, since the area consumed by the design is considered by the number of CLBs and the number of input/output blocks (IOBs). In recent years, serial-serial processors have received considerable attention as an alternative approach to conventional parallel and serial-parallel architectures, particularly in the area of signal processing [13-15]. A major advantage offered by bit-serial processors over their counterparts resides mainly in the low number of I/O pins and in the more efficient use of chip area, which results from a use of a minimum number of gates and interconnections between components. Serial-serial multiplication techniques have been in use for many years [13-15]. However, all reported structures have been restricted to bit serial-serial approaches, which process one bit at a time and hence are only suitable for low speed applications.

The concept of digit serial-serial has not been addressed so far in literature. As opposed to the serial-parallel structures, in digit serial-serial architectures both input data are fed in serial fashion (i.e. one digit from each operand enters the processor during one cycle). This will reduce the number of I/Os from 2n+N in the case of the digit serial-parallel multipliers to 3n in the case of digit serial-serial multipliers, where $N=K\times n$ is the worldlength, n is the digit size and K is the number of digits per word.

In this paper, the design of digit serial-serial multipliers based on the radix-2^n arithmetic is presented. One radix-2^n serial-serial architecture is derived and a new cell for digit serial-serial multiplication is introduced. Also, an area efficient digit serial-serial multiplier using the technique previously proposed by the authors in [14] for the bit serial-serial structures is presented. Architectures based on digit serial-serial structures offer to find the best design approach needed in order to find, for any application, the best tradeoffs between cost, speed, efficient area utilisation, throughput, I/O pin limitation and power dissipation.

2. Design of the digit serial-serial multiplier

The radix-2^n algorithm can be considered as a generalisation of the binary arithmetic where the simplest building block is an AND gated Full Adder [8]. Using the radix-2^n arithmetic and assuming unsigned numbers, two N-bit numbers, X and Y, can be divided into K digits of n-bit each and can be written as

$$X = \sum_{i=0}^{K-1} X_i 2^{in} \qquad\qquad Y = \sum_{j=0}^{K-1} Y_j 2^{jn} \tag{1}$$

where X_i represent the i^{th} digit of the X and Y_j is the j^{th} digit of Y. The product, P, of two N-bit numbers, X and Y, can be written using the radix-2^n arithmetic as

$$P = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} X_i Y_j 2^{(i+j)n} \tag{2}$$

Equation 2 can be computed using the dependency graph shown in figure 1 for K=4 where each node performs the multiplication of two n-bit digits, X_i and Y_j, and adds the product to two other n-bit digits, S_{in} and C_{in}. The result will always be a 2-digit number, the most significant digit (MSD) is the carry digit, C_{out}, and the least significant digit (LSD) is the sum digit, S_{out} [6-8]. To

obtain digit serial-parallel structures, the dependency graph in figure 1, is projected along either the *i* or *j* directions. In this case one of the axis defines time while the other define space.

In serial-serial computation, at any one cycle the data is entered as a pair of digits, (X_k,Y_k), having the same significance, *k*. The products, X_kY_k, are computed at the nodes on the diagonal line $i=j$. In the index space (i, j), $(i+j)$ represents a family of lines perpendicular to the line $i=j$ and determines the significance. In serial-serial computation, the family of lines $k=i+j$ should also define time. The reason is that, at every cycle a new pair of digits (X_k, Y_k) enters the computation along one of the lines $(i+j)$. As a result, no single index is used alone to define either time or space. Both indices *i* and *j* are used to define time and hence a single projection along either *i* or *j* index will not result in digit serial-serial architecture.

Using a new space coordinates (k, l) defined as:

$$k = i + j \qquad\qquad l = i - j \tag{3}$$

Equation 2 can be rewritten as

$$P = \sum_{k=0}^{2(K-1)} \sum_{l=-(K-1)}^{K-1} X_{\left(\frac{k+l}{2}\right)} Y_{\left(\frac{k-l}{2}\right)} 2^{kn} \tag{4}$$

It is clear from equation 3 that the only valid values for *k* and *l* are those which result in $(k+l)/2$ and $(k-l)/2$ to be integer values. From figure 1 and equation 4, it can easily be seen that in serial-serial computation where both input data *X* and *Y* are treated similarly, the partial products generation is carried out in the same manner symmetrically across the *k* axis. Any node $(k, |l|)$ performs the same operation as a node $(k, -|l|)$ for $l \neq 0$. As a result, a radix-2^n serial-serial multiplication can be defined by the following equation

$$P = \sum_{k=0}^{2(K-1)} \left[X_{\frac{k}{2}} Y_{\frac{k}{2}} + \sum_{l=1}^{K-1} \left(X_{\left(\frac{k+l}{2}\right)} Y_{\left(\frac{k-l}{2}\right)} + X_{\left(\frac{k-l}{2}\right)} Y_{\left(\frac{k+l}{2}\right)} \right) \right] 2^{kn} \quad (5)$$

Each of the partial products in equations 4 and 5 is a 2*n*-bit number, which can be divided into least significant *n*-bit number, LSD, and most significant *n*-bit number, MSD. As reported in [8], to carry out the radix-2^n computation, it is necessary that either the LSD or the MSD are moved from nodes with significance, *k*, to nodes with significance (*k*-1) or (*k*+1), respectively. The dependency graph in figure 1 takes into account the transfer of the MSD from nodes (*k*, *l*) to nodes (*k*+1, *l*+1), where the carry digit, C_{out}, is made up of the addition of the MSD with the carries generated from the addition of the LSD with the carry digit, C_{in}, and the sum digit, S_{in} [8].

A dependency graph, which allows computation of the radix-2^n serial-serial multiplication as described by equation 5 is shown in figure 2 for K=4. In this case the carry digit, C_{out}, is made up of the addition of the two MSDs from the two partial products in equation 5 with the carries generated from the addition of the LSDs with the carry digit, C_{in}, and the sum digit, S_{in} [8]. Again the transfer of the carry digits is carried out from nodes (*k*, *l*) to nodes (*k*+1, *l*+1).

To obtain digit serial-serial architectures, the dependency graph in figure 2 is projected onto the line $l = 0$ (i.e. onto the k axis) in the direction $[1, 1]^T$. The resulting radix-2^n serial-serial multiplier is shown in figure 3 for K=4. It consists of K basic cells, which corresponds to the mapping of all nodes (*k*, *k*-2*p*) with $k \geq 2p$ onto a single node (2*p*, 0), where $p = 0, 1, 2, \ldots, (K-1)$. The node (2*p*, 0) is then termed cell, *p*. Due to the chosen projection, it can be seen that the carry digit, C_{out}, generated by cell, *p*, is delayed then fed back to the same cell. It can also be seen that during cycle

$k=p$, a pair of input data, X_p and Y_p, is broadcasted to the cells 0 to p, and is stored in the cell, p, for further computations. A cell, p, start by computing the product X_pY_p during cycle $k=p$ and then the terms $[X_pY_{k+p}+X_{k+p}Y_p]$ are computed during the next K-p-1 cycles. The terms computed in cell p will be added to an accumulating result from the cell, p+1, and then propagated in the next cycle to the cell, p-1. After the K^{th} cycle, cell p is used to shift the partial results from left to right.

Following the above discussion, a basic cell, p, should contain two n-bit multipliers, accumulators and latches controlled by a control signal, *Control*, to allow storage of the two input data X_p and Y_p. The architecture of the basic cell is shown in figure 4. Carry save arithmetic implemented using n-bit 4-2 compressors is used for the accumulation of the partial results to reduce the hardware cost. Each n-bit 4-2 compressor is made of two n-bit carry save adders (CSAs). The basic cell is subdivided into two stages working in a pipeline manner. The first stage deals with the multiplication process of the relevant data, while the second stage performs the addition of the result of the first stage with the shifted result from the neighbour cell. The first part of the first stage of the basic cell is implemented using two partial product generators and $\log_2(n)$ rows of n-bit 4-2 compressors arranged in a tree type structure as shown in figure 4 for n = 4 and can be generalised for any digit-size. The carry digit, C_{out}, consist of the MSDs of the two partial products, X_pY_{k+p} and $X_{k+p}Y_p$, and all the carry bits generated by the n-bit 4-2 compressors. The carry digit, C_{out}, is delayed and then fed back into the same cell as shown in figure 3. This is carried out by the delay and then the feedback of all the carry bits from the n-bit 4-2 compressor and the MSDs of all the partial products, X_pY_p, X_pY_{k+p} and $X_{k+p}Y_p$. The feed back of the two carry-out bits from the most left 1-bit compressor within each of the n-bit 4-2 compressors is achieved by storing these bits to be added in the next cycle as shown in figure 5. It is worth noting that feeding these bits down to the least significant bit position of the next row of CSAs is equivalent to the feedback of these bits to the

LSB position of the same row of CSAs. The delay and feed back to the same cell, p, of the MSDs of all the partial products, X_pY_p, X_pY_{k+p} and $X_{k+p}Y_p$, is carried out by delaying these MSDs and adding them to the LSDs of the same partial products generated in the following cycle. This is achieved by delaying the input data, Y_p (during cycle $k=p$) or Y_{k+p} and X_{k+p}, (during cycles $k \in [p+1, K-p]$) prior to their multiplication with the stored data X_p and Y_p, as shown in figures 6 and 7. This is made possible by the fact that the during the computation of the these partial products one of the operands, i.e X_p in the case of the products X_pY_p and X_pY_{k+p} and Y_p in the case of the products $X_{k+p}Y_p$, is stored permanently in cell, p. This results in the reduction of the number of delay elements from $n(n-1)/2$ to $(n-1)$ per partial product generator. This is a total of $K(n-1)(n-2)$ reduction in the total number of delay elements. The multiplexers in the partial product generator (PPG 1) in cell, p, are required to allow the computation of the product X_pY_p as well the storage of X_p during cycle $k=p$.

The partial product generated by the first stage is shifted to the second stage, which consists of a n-bit 4-2 compressor. Due to the use of the carry save arithmetic, the sum digit, S, is made up of two digits, S_1 and S_2. The second stage is used to add the carry and sum word from the first stage to the partial sums, S_{1in} and S_{2in} transferred from the neighbouring cell on the left. The output of the second stage is shifted to the next cell on the right and the process is repeated. A final adder is required to sum the two digits, S_{1out}, S_{2out} produced by the last cell on the right hand side of the multiplier to obtain the correct result as shown in figure 8. This can be implemented using the bit-level pipelined digit-serial adder proposed by the authors [6-8].

A 32-bit radix-2^n serial-serial multiplier has been implemented using the S40BG256 device from the SPARTAN family. The area, time and area-time complexities for the 32-bit area efficient radix-2^n serial-serial multiplier as a function of the digit size, n, are shown in figures 9, 10 and 11, respectively. It can be seen that as the digit size, n, increases both the hardware cost in term of the number of gates and the throughput rate are increased. However, the area-time complexity shows that there is a digit size, in this case n=4, which gives the optimum area-time performance.

It is worth mentioning that the proposed radix-2^n serial-serial architecture can be pipelined to the bit level (in this case the cycle time is that of an AND gated FA). This is made possible by the fact that all the data within the basic cell move in the same direction (from top to bottom). The feed back of the carry digit is carried out by delaying it by one cycle and shifting it to the next lowest digit significance as shown in figures 5, 6 and 7.

3. Design of the area efficient digit serial-serial multiplier

By projecting the dependency graph of the radix-2^n serial-serial multiplier shown in figure 2 into the direction $[1, 1]^T$, it can be seen that K-p nodes are mapped into a single cell p. As a result, the pth cell, performs 2(K-p)-1 multiplications. This implies that the cell with p=0, performs 2K-1 multiplication, while the cell with p=K-1, performs only one multiplication. Hence moving from cell p=0 to cell p=K-1, the cells usage become less and less. It can also be seen that after the K^{th} cycle, the last part of the multiplication process is spent in generating carries and shifting out results. Also, the further a cell is from the output of the array, the more time it spends being idle. This indicates that the area utilisation of the multiplier obtained in the previous section is very inefficient.

To obtain an area efficient digit serial-serial multiplier, the dependency graph shown in figure 2 is modified. From figure 2, It can be seen that the partial sums generated by nodes (k, k-2p) with $k \geq 2p$ and $p \geq$ K/2 are added to the carry digits generated by nodes (k, k-2p) for p K/2-1. It can also be seen that the nodes (k, k-2p) with $k \geq$ 2K-1 and p K/2 -1, perform only the accumulation and shift of the partial results. Therefore, it is possible to move all the nodes (k, k-2p) with $k \geq 2p$ and $p \geq$ K/2 to the nodes (k, k-2p) with $k \geq$ 2K-1 and p K/2 -1 as shown in figure 12. This process requires the input data (X_p, Y_p) with $p \geq$ K/2 to be fed again into the array as shown in figure 12. The area efficient radix-2^n serial-serial multiplier, which is shown in figure 13, is obtained by projection of the dependency graph in figure 12 in the direction $[1, 1]^T$. The resulting serial-serial multiplier performs the multiplication using K/2 cells (i.e. half the number of cells required by the multiplier in figure 8). The architecture of the basic cell is the same as that used in the multiplier shown in the previous section and is shown in figure 4. However, two K/2 shift register are required to store the K/2 most significant digits of the data words X and Y. The stored data should start entering the multiplier at the $(K+1)^{th}$ cycle. This can be achieved by employing two multiplexers. The signal controlling the multiplexer is derived from the control signal, *Control*, by an EXOR-gate as shown in Figure 13, which will allow each of the data word followed by its most significant digits to be fed to the multiplier. As a result of the re-mapping of the input data, a significant saving of about 50% of the hardware requirement can been achieved.

4. Performance Analysis

In this section the performance of the proposed area efficient digit serial-serial multiplier is compared to that of the digit serial-parallel multiplier which was first reported in [6] and also

reported in [9-10]. The throughput rates of both multipliers are comparable. However, the proposed digit serial-serial multiplier is slightly slower due to the existence of the multiplexers within the data paths (see figure 6). Both multipliers with N=32 have been implemented using the S40BG256 device from the SPARTAN family. Figure 14 shows the area occupied in terms of the number of gates by each multiplier as a function of the digit size n. It can be seen from figure 14 that the area efficient digit serial-serial multiplier has a much lower hardware cost than the digit serial-parallel multiplier. It was noted that reductions in the hardware costs of 34% for n=2 to 47% for n=16 have been achieved. From figure 10 and figure 14, it can also be seen that similar reduction in the hardware cost have been achieved by the area efficient digit serial-serial multiplier when compared to the original digit serial-serial multiplier discussed in section 2. It is worth mentioning that the gate count in figures 10 and 14 does not include the additional number of gates required for input/output blocks (IOBs).

5. Conclusions

In this paper the design of digit serial-serial multipliers is presented. The design procedure is based on the radix-2^n arithmetic, which is a generalisation of the binary arithmetic. An equation for the radix-2^n serial-serial multiplication and its corresponding dependency graph have been derived. An architecture of a radix-2^n serial-serial multiplier is obtained by a projection of the dependency graph. The radix-2^n serial-serial multiplier has been implemented using the S40BG256 device from the SPARTAN family to prove functionality and to assess its performance in terms of area and time with respect of the digit size. Furthermore, a dependency graph and hence an architecture for an area efficient radix-2^n serial-serial multiplier has been derived. The hardware cost in term of the number of gates (excluding the number of gates required for IOBs) of the area efficient radix-2^n

multiplier has been compared to a digit serial-parallel multiplier. It was found that a saving of up to 47% in the hardware cost has been achieved by the proposed area efficient radix-2^n serial-serial multiplier. This on the top of the reduction in hardware cost associated to the number of IOBs due the lower number of I/Os required by the proposed digit serial-serial multiplier.

The proposed architectures can be pipelined to the bit level to increase the throughput rate. Hence, they offer the designer the increased flexibility of finding the best trade off between throughput rate and hardware cost by varying the digit size and the number of pipelining levels.

References:

1. P.B. Denyer and D. Renshaw: 'VLSI Signal Processor - a Bit-Serial approach' Addison-Wesley, 1985.

2. S.Y. Kung: 'VLSI Array Processors' Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

3. S.G. Smith, M.S. McGregor and P.B. Denyer: “Techniques to Increase the Computational Throughput of Bit-Serial Architectures”, IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP, pp. 543-546, Apr. 1987.

4. K.K. Parhi: “A Systematic Approach for Design of Digit-Serial Signal Processing Architectures”, IEEE Trans. Circuits and Systems, **38**, pp. 358-375, April 1991.

5. R. Hatley and P. Corbett: “Digit-Serial Processing Techniques”, IEEE Trans. Circuits and Systems, **37**, pp. 707-719, June 1990.

6. A. Ashur, A. Aggoun and M.K. Ibrahim: “Systolic digit-serial multiplier”, IEE Proceedings-Circuit Devices Syst., **143**(1), pp. 14-20, Feb. 1996.

7. A. Aggoun, M.K. Ibrahim and A. Ashur: “A Novel Cell Architecture for High Performance Digit-Serial Computation” Electronic Letters, **29**(11), May 1993.

8. A. Aggoun, M.K. Ibrahim and A. Ashur: “Bit-level pipelined digit-serial array processors”, IEEE Transaction on Circuit and Systems (CAS), **45**(7), pp. 857-867, July 1998.

9. A. Aggoun, M.K. Ibrahim and A. Ashur: “Design methodology for sub-digit pipelined digit-serial IIR filters”, Journal of Signal Processing, **68** (1), pp. 73-86, July 1998.

10. Y.N. Chang, J. Satyanarayana and K.K. Parhi, “Low-Power Digit-Serial Architectures”, Proc. of IEEE Int. Symp. on Circuits and Systems , pp. 2164-2167, Hong Kong, June 1997.

11. Y.N. Chang, J.H. Satyanarayana and K.K. Parhi, “Systematic Design of High-Speed and Low-Power Digit-Serial Multipliers”, IEEE Trans. on Circuits and Systems, Part II: Analog and Digital Signal Processing, **45**(12), pp. 1585-1596, Dec. 1998.

12. A. Stone and E. S. Manolakos, “DG2VHDL: A Tool to Facilitate the High Level Synthesis of Parallel Processing Array Architectures”, Journal of VLSI Signal Processing, **24**(1), pp. 99-120, Feb. 2000.

13. P. Ienne and M. A. Viredaz: 'Bit-serial multipliers and squarers', IEEE Trans. Computer, **43**(12), pp.1445-1450, 1994,.

14. A. Aggoun, M.K. Ibrahim and A. Ashur: ‘Area-time efficient serial-serial multipliers’ Proc. of IEEE Int. Symp. on Circuits and Systems, **5**, pp. 585-588, May 2000.

15. O. Nibouche, A Bouridane, and M. Nibouche: ‘New architectures for serial-serial multiplication’ Proc. of IEEE Int. Symp. on Circuits and Systems, **2**, pp. 705-708, May 2001.

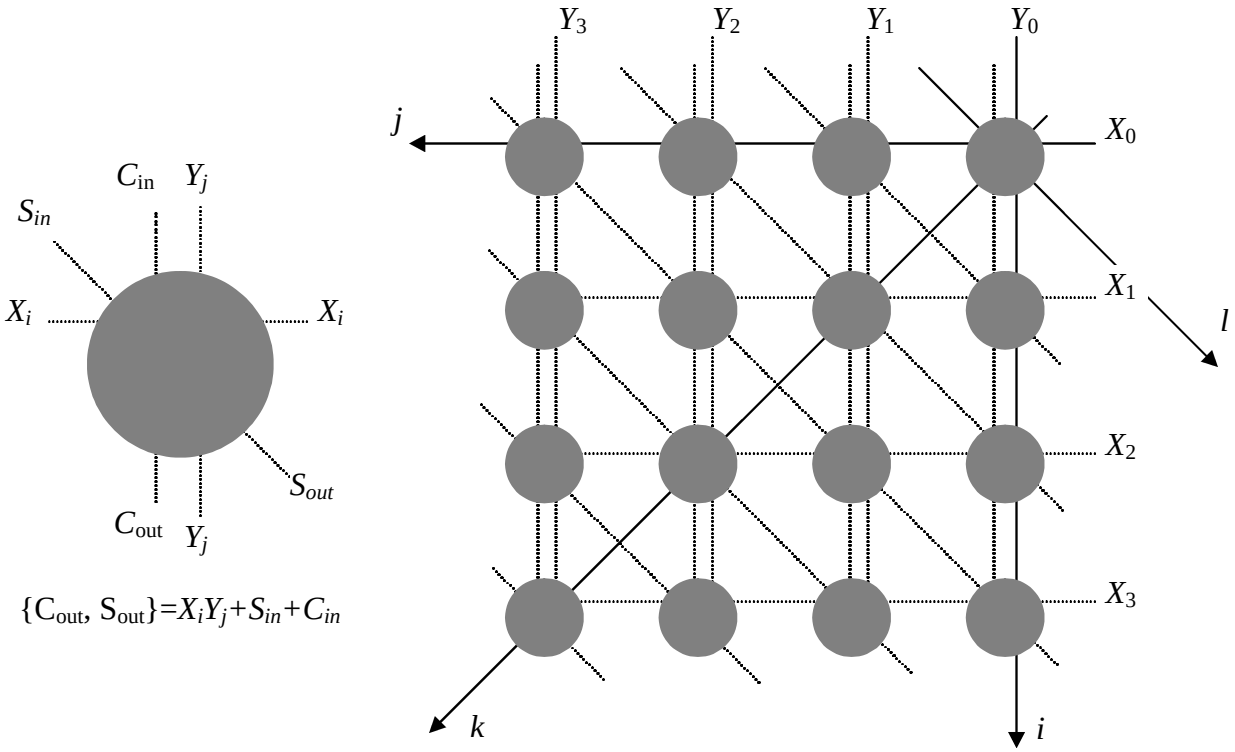

Figure 1: Dependency Graph of a radix-2^n multiplication for K=4 [8]

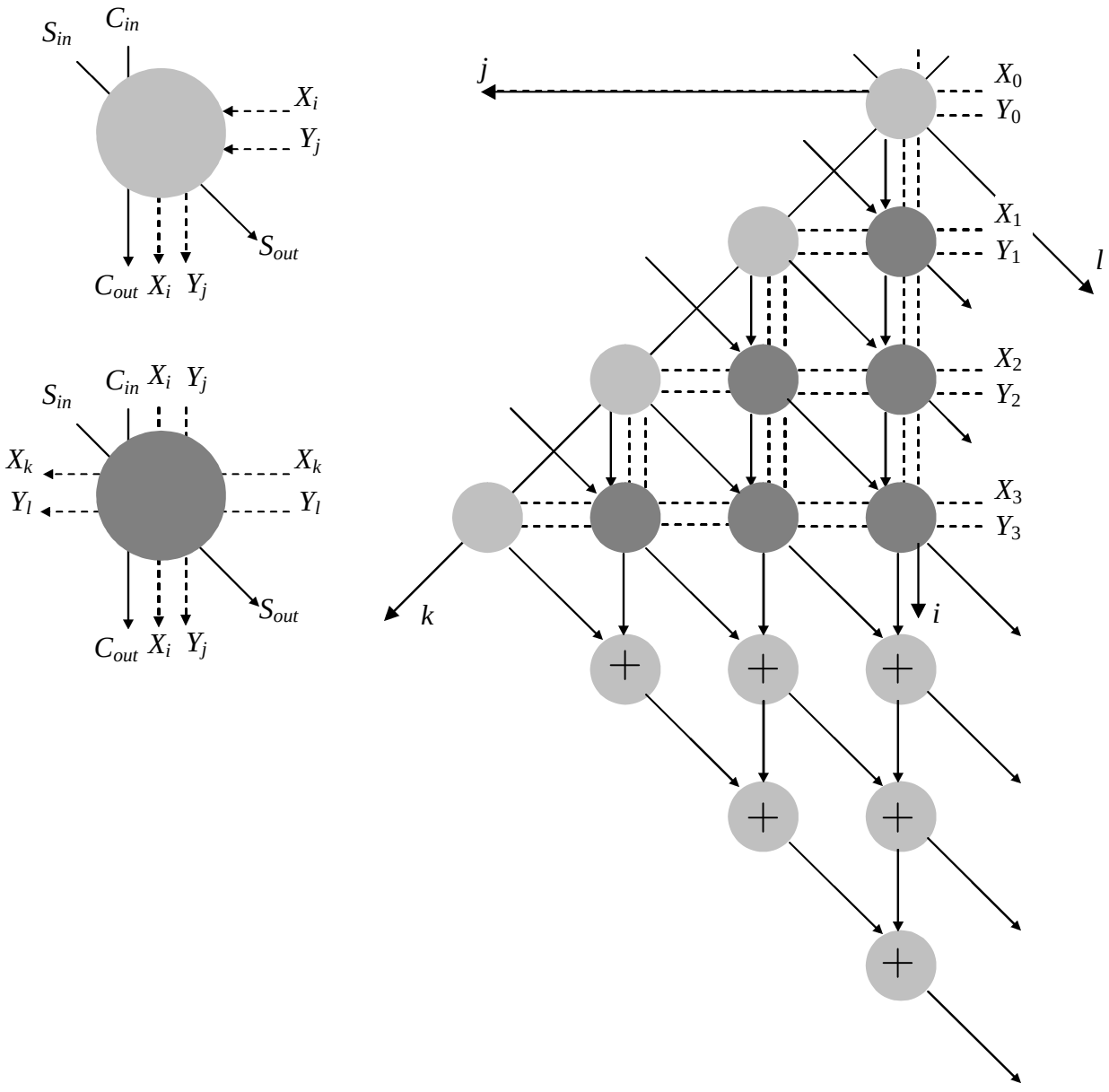

Figure 2: Dependency Graph of a radix-2^n serial-serial multiplication for K=4.

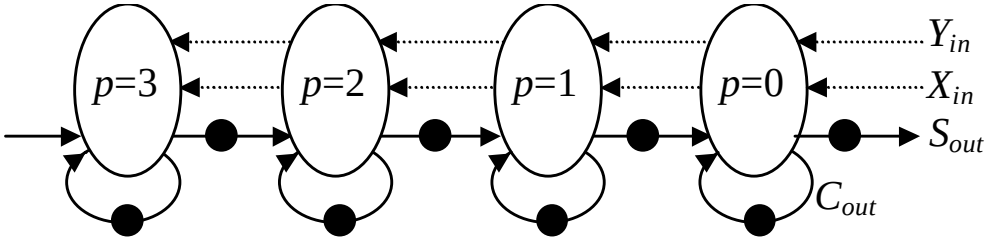

Figure 3: Digit serial-serial architecture obtained by projection of the DG in figure 2 in the direction $[1, 1]^T$.

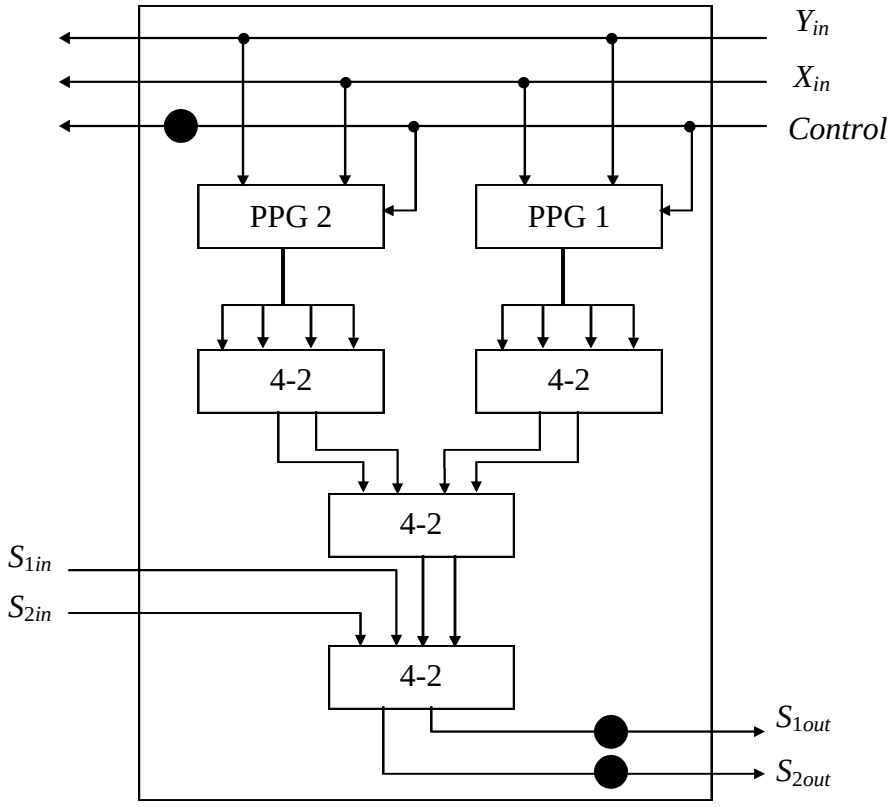

Figure 4: Basic cell of the radix-2^n serial-serial multiplier

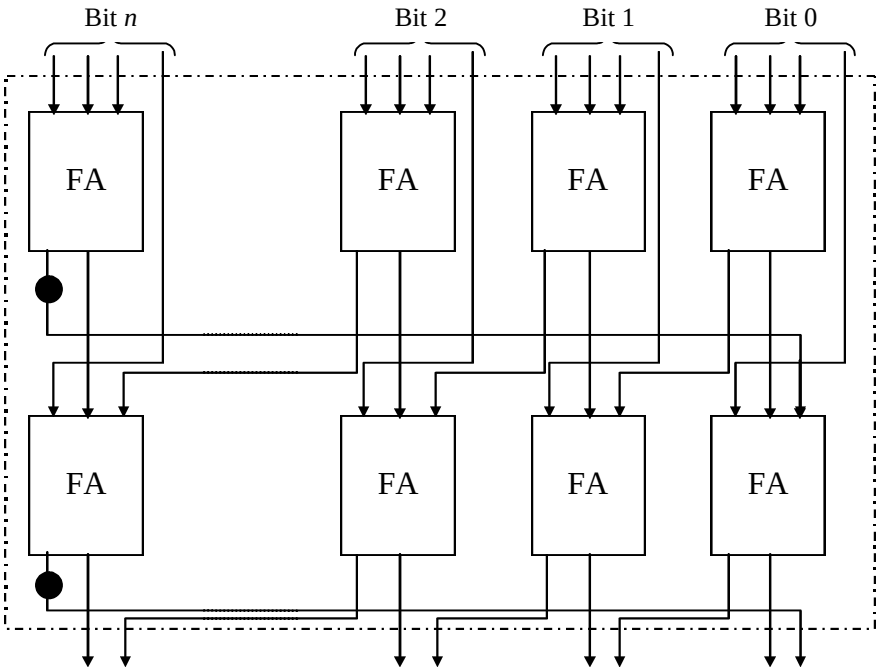

Figure 5: *n*-bit 4-2 compressor unit.

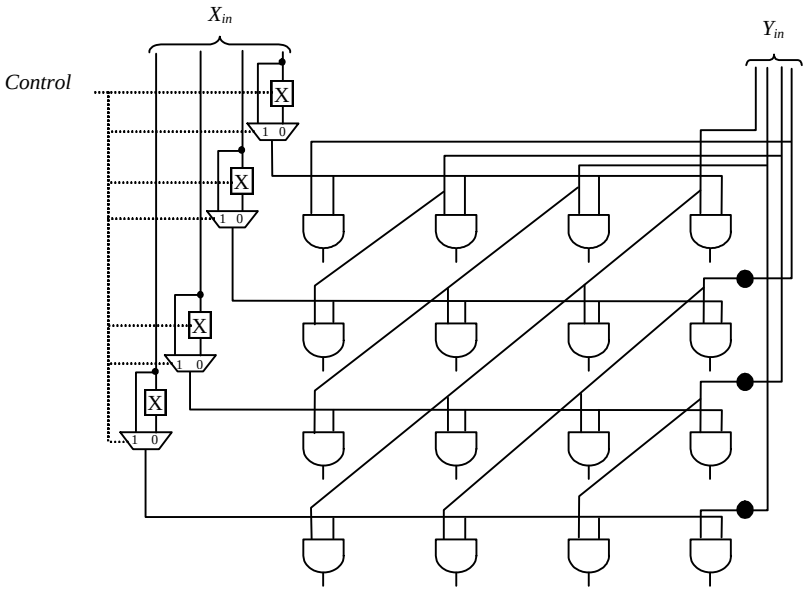

Figure 6: Partial product generator 1 (PPG 1).

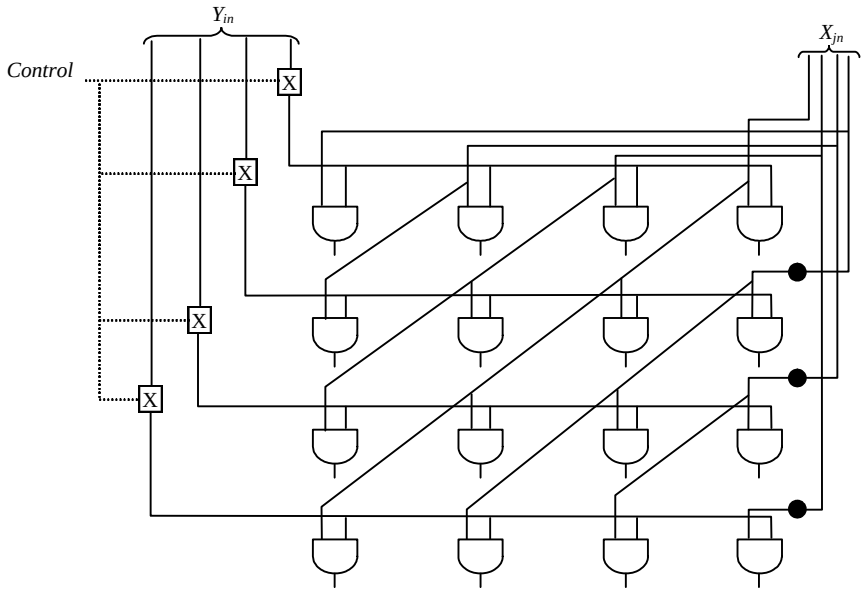

Figure 7: Partial product generator 2 (PPG 2).

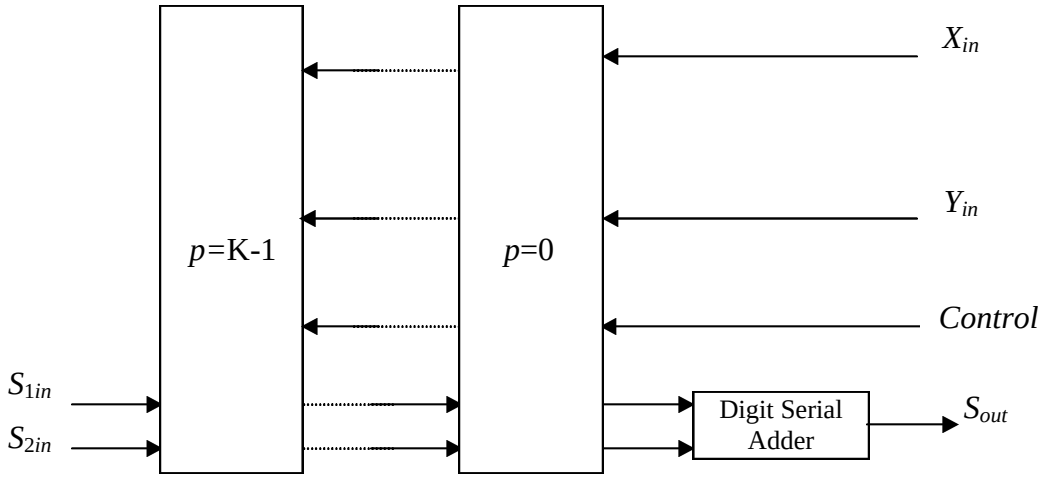

Figure 8: Radix-2^n Serial-Serial Multiplier

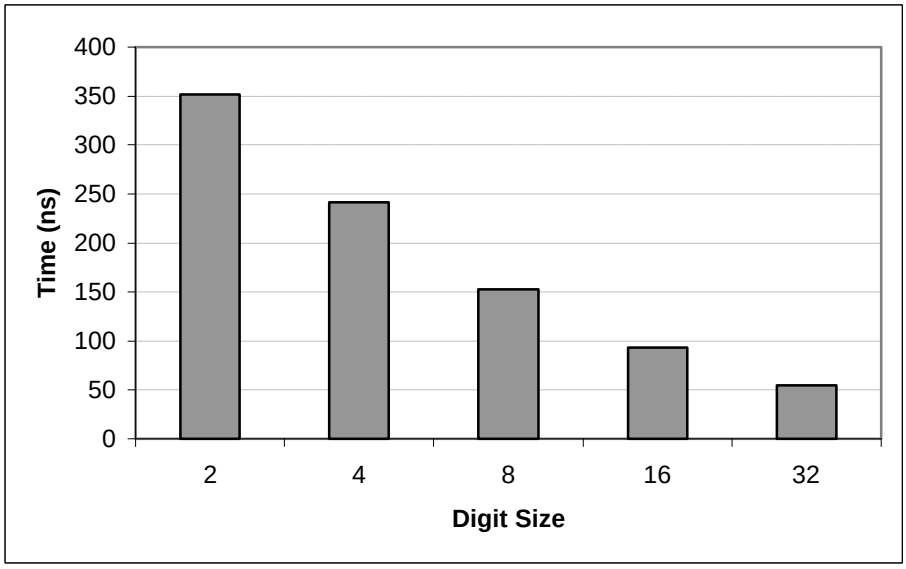

Figure 9: Time as a function of the digit size for a 32-bit area efficient digit serial-serial multiplier.

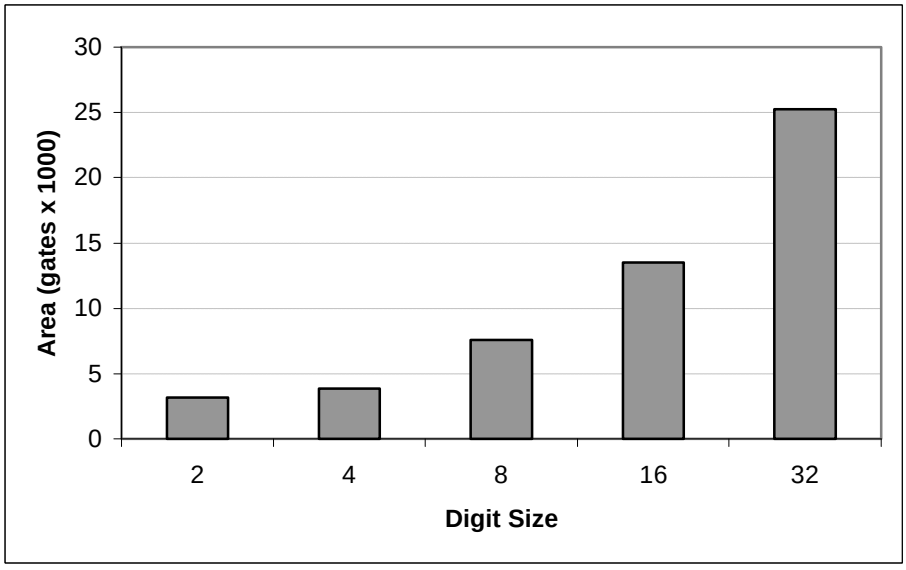

Figure 10: Area as a function of the digit size for a 32-bit area efficient digit serial-serial multiplier.

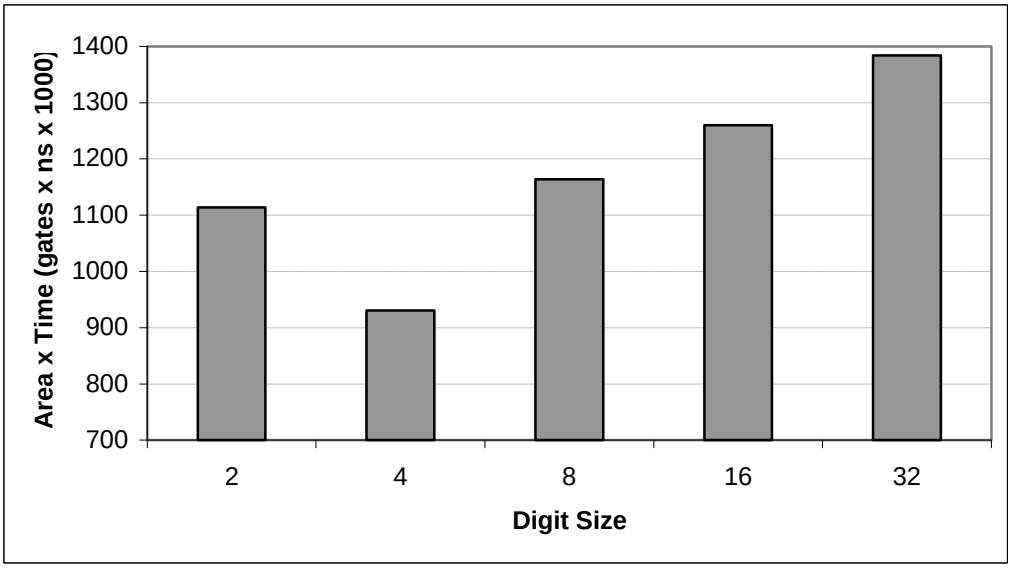

Figure 11: Area × Time as a function of the digit size for a 32-bit area efficient digit serial-serial multiplier.

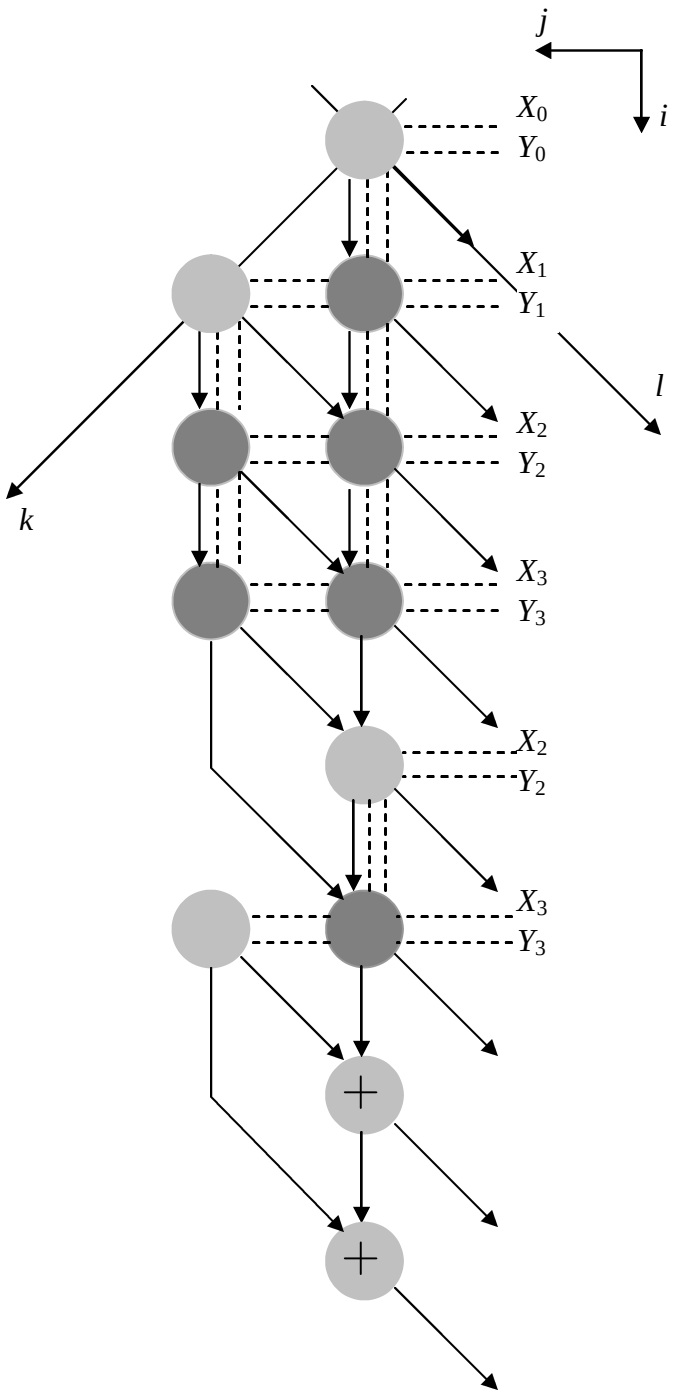

Figure 12: Dependency graph of the area efficient radix-2^n serial-serial multiplier.

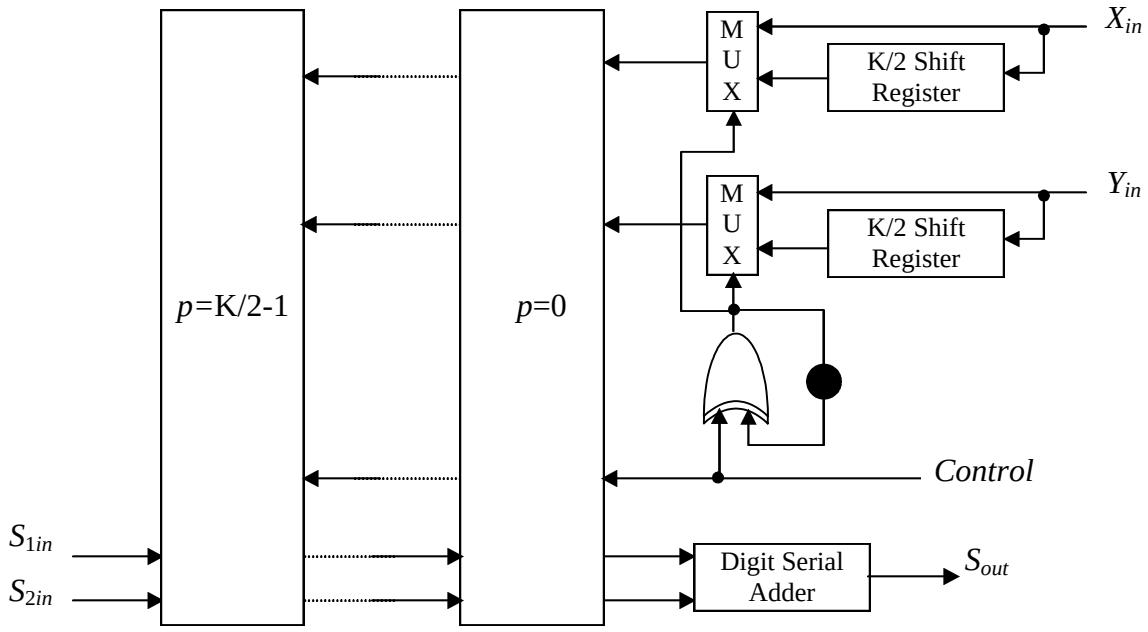

Figure 13: Area efficient digit serial-serial multiplier

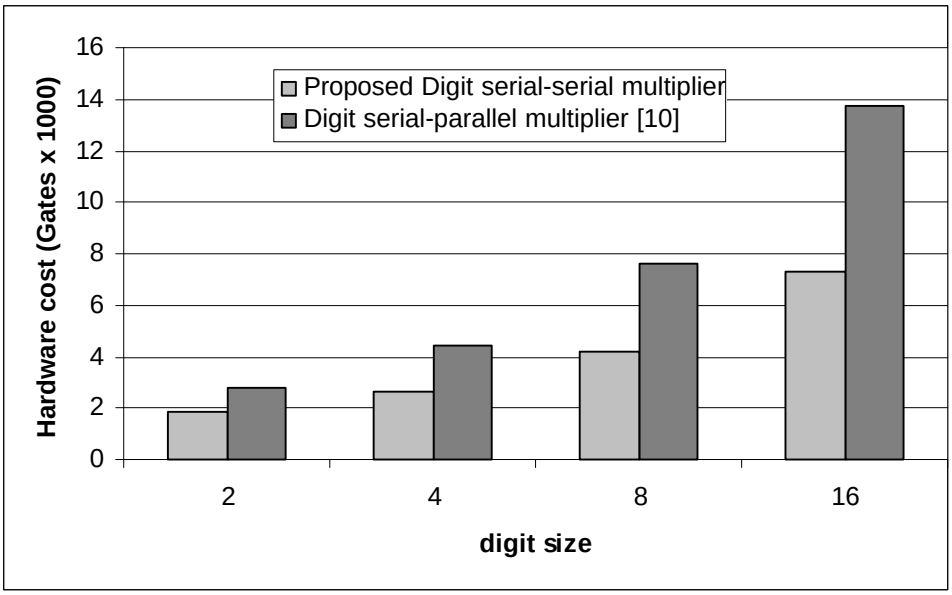

Figure 14: Hardware cost of the 32-bit area efficient Digit serial-serial and Digit serial-parallel multipliers as a function of the digit size.