Journal of VLSI Signal Processing, 4, 199–212 (1992)
© 1992 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Power Dissipation of VLSI Array Processing Systems*

PAUL M. CHAU AND SCOTT R. POWELL
Department of Electrical and Computer Engineering R-007, University of California San Diego, La Jolla, CA 92093-0407

Received January 20, 1991; Revised manuscript received October 20, 1991.

Abstract. The Power Factor Approximation (PFA) power estimation method is reviewed and applied to VLSI array processing systems. The power dissipation of 1, 2, and 3 dimensional algorithms implemented on linear, hexagonal, and cubic processor arrays is investigated. Closed form equations are developed which show how the overall power dissipation is influenced by an algorithm's size and dimensionality, the target array processor's size and dimensionality, and the adopted partitioning strategy. The power estimation methods developed in this paper can be applied in the early phases of VLSI algorithm/architecture design, selection, and partitionment. The power dissipation of a matrix-matrix multiplication operation is estimated as an example application.

1. Introduction

There has been much interest over the past decade in developing algorithms suitable for VLSI implementation. VLSI architectures capable of implementing these algorithms such as the systolic array [1], the wavefront array [2], and multiprocessors [3] have been developed to serve as a general methodological guideline for the design of special purpose array processing systems. Computational complexity theorists have proposed the use of graph theoretical techniques to derive asymptotic bounds on the area and time of such systems [4]. Partitioning strategies have been developed to map computations of large size problems to processor arrays of a smaller size using the metrics of area and speed [5]–[7]. However, these area-time approaches neglect power dissipation. Since power dissipation is often what limits the level of integration possible within a VLSI technology, accurate figures of merit for VLSI algorithms and architectures should account for area, speed, and *power* [8].

Ward [9] and Thompson [10] suggest characterizing a technology in terms of watts per gate-Hz or Jouls per wire and then multiplying by an algorithm's number of *equivalent* gates and speed to estimate the overall power. This technique permits a desirable separation between technology dependent and independent factors. However, this fairly course-grained measure does not readily account for the effects of a circuit's functional characteristics, alternative design styles (e.g., synchronous vs. asynchronous), or advancing technologies. We have previously presented the Power Factor Approximation (PFA) technique [11] for estimating the power dissipation of special purpose VLSI chips. As with the approaches of [9] and [10], algorithmic/architectural dependent influences on power dissipation are separated from technology specific factors but the PFA method also accounts for I/O power, gate activity, and design style.

In Section 2 of this paper, we review the PFA technique leading to power dissipation equations for memory, I/O, and multiplication. In Section 3, we use the PFA method to determine the power dissipation of computational algorithms which have been partitioned onto linear, hex, and cubic processor arrays, Although results are derived for specific arrays and partitioning schemes, the power estimation method presented can easily be adopted for arbitrary arrays and partitioning schemes. In Section 4, we apply the results obtained in Sections 2 and 3 to a 15 × 15 matrix multiplication example implemented on an eight processing element linear, hex, and cubic array processor.

2. Power Dissipation in VLSI Chips

The most common VLSI technology is CMOS primarily because its scaling properties and low power dissipation permit greater levels of integration than alternative technologies [10]. Since complementary CMOS logic gates require power only during signal transitions, static power dissipation is nearly eliminated. Detailed

*This work was supported in part by the Hughes Aircraft Company fellowship program and the NSF initiation grant MIP-99-10437.

discussions of the power dissipation of basic CMOS logic gates can be found in [12]–[15]. Calculating the power dissipation of large CMOS circuits or entire VLSI chips is a difficult task and most current approaches involve logic or switch level simulations of the entire chip [16]–[18]. However, these methods can only be appropriately applied after a system has been designed down to the gate level. The large amount of time involved in synthesizing a gate level logic description and performing the computer simulations severely limits the number of alternatives which can be economically evaluated during the design phase of a VLSI system. The power factor approximation (PFA) technique [11] permits the estimation of power dissipation early in the design cycle. The PFA power estimation method can be used at the algorithmic and architectural design level and is the method adopted in this paper for estimating the power dissipation of processing elements and complete VLSI array processors. In this section, we review both the basic mechanisms of CMOS power dissipation and the PFA power estimation technique.

2.1. CMOS Power Dissipation

The simplest CMOS logic element is the inverter. As shown in figure 1, there are two main currents which flow each time the inverter's output transitions. These currents are called the dynamic, $i_{dy}(t)$, and short circuit, $i_{sc}(t)$, components. The dynamic component is from charging and discharging the load capacitance C_L and the short circuit component is from n- and p-channel transistors conducting simultaneously as the input is transitioning. In general, any complementary logic element will exhibit the same behavior. Note we are assuming that the power dissipation due to static leakage current (proportional to transistor size) and circuit resistances (I^2R) is small in comparison to the dynamic and short circuit power dissipation.

For periodic signals, the average power dissipation for a single logic element as in figure 1 is defined as:

$$P_{avg} = \frac{1}{T}\int_0^T V_{dd} i_{dd}(t)\, dt$$
$$= V_{dd} f \int_0^T i_{dd}(t)\, dt = V_{dd} f q \qquad (1)$$

where

T = period

$f = 1/T$ = frequency

q = the charge transferred per cycle

For each logic transition, the charge transferred to the load capacitor by the dynamic current is $q = C_L V_{dd}$. For nonperiodic switching, the average power dissipation of a single CMOS logic element can be found by defining an activity factor:

A_f = average number of switchings per cycle $(0 \leq A_f \leq 1)$

and thus the power dissipated by a single logic element is given by:

$$P_{avg} = A_f \left[C_L V_{dd}^2 f + V_{dd} f \int_0^T i_{sc}(t)\, dt \right] \qquad (2)$$

where

$i_{sc}(t)$ = the short circuit component

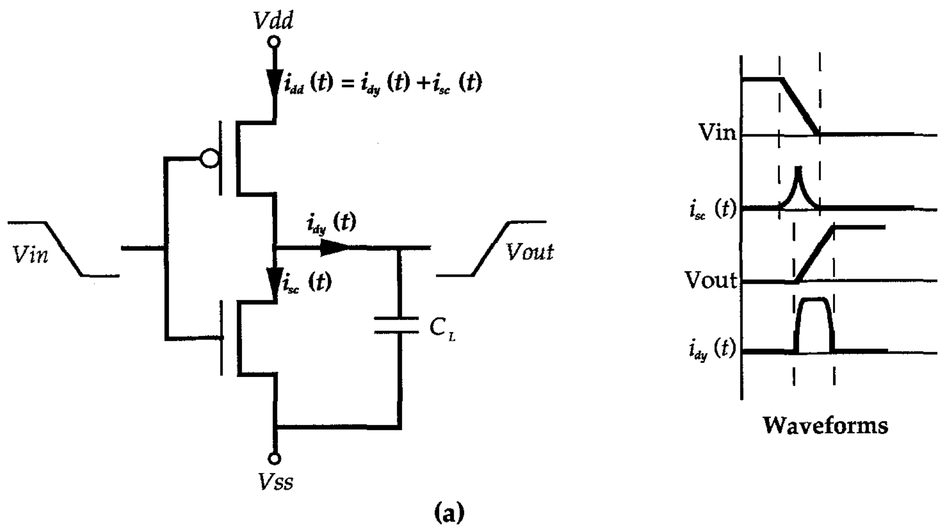
$C_L V_{dd} f$ = the dynamic current component, $i_{dy}(t)$

A thorough discussion and analysis of short circuit power dissipation in CMOS inverters can be found in [14]. Yacoub [13] presents a method for separating the dynamic and short circuit components of general multigate CMOS circuits and finds that the short circuit component can range from 20%–50% of the total power dissipation.

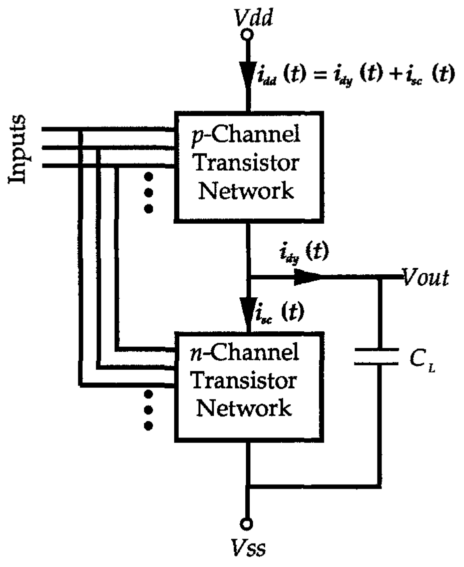
2.2 The Power Factor Approximation Method

The PFA approach is a method for estimating the power dissipation of large functional blocks in terms of high level parameters. The goal is to permit exploration of algorithmic, architectural, and technological alternatives early in the VLSI system design cycle. The PFA power estimation method separates the influences of technology dependent factors from technology independent factors and accounts for I/O power, design style, and the dependence on gate activity.

The PFA concept is based on the assumptions: (1) for a given functional block (e.g., a multiplier or RAM) the average activity factor and short circuit current *per logic element* are determined mainly by the technology and the type of functional block and not by the size of the functional block; (2) the power dissipated by a CMOS gate is directly proportional to the rate of operation (see equation (2)); and (3) for a given functional block operating at a processing rate f, the overall power dissipation can be determined by summing the power dissipation of each individual gate. Further, assuming there are only local interconnections between logic elements (i.e., no broadcasting of signals), the load capacitance *per logic element* will not be a function



(a)


Vdd
$i_{dd}(t) = i_{dy}(t) + i_{sc}(t)$
Inputs
p-Channel Transistor Network
$i_{dy}(t)$
Vout
$i_{sc}(t)$
n-Channel Transistor Network
C_L
Vss

(b)

Fig. 1. Dynamic and short circuit current components in (a) a simple CMOS inverter (b) a general multi-input CMOS logic gate.

of the size of the functional block. Based on these assumptions, the overall power dissipation of a functional block can be put into the equivalent form of (see [11] for further explanation):

$$P_{avg} = \kappa G f \tag{3}$$

Using (2) and (3), the PFA constant κ is given by:

$$\kappa = \frac{\sum_{i=1}^{G} A_{f_i} C_{L_i} V_{dd}^2}{G} + \frac{\sum_{i=1}^{G} A_{f_i} V_{dd} \int_0^T I_{sc_i}(t)\, dt}{G} \tag{4}$$

Where $G \equiv$ the number of logic elements.

Thus, the PFA constant κ is equal to the average power dissipated per gate per processing cycle specific to a given type of functional block (i.e., an application specific power-delay product). It is usually more convenient to use a higher level parameter than the number of gates for G. For example, in cases where the number of gates is proportional the word length G could be defined as the number of bits/word giving κ the units of Watts/bit-Hz.

Closed form equations for the power dissipation of integer multipliers, static memories, and I/O operations based on the PFA power estimation technique are presented in [11]. Also presented are the current values of the associated PFA constants for 1 μm CMOS based on reported power dissipation. These results are summarized below as reference for their application in subsequent sections:

Integer Multiplier: $P_{mult} = \kappa_{mult} Q^2 f_{mult}$ (watts)
($Q \times Q$ Bit) $\kappa_{mult} \approx 15$ pW/bit^2-Hz (5)

Static Memory: $P_{mem} = \kappa_{mem} WQf_{mem}$ (watts)
(W Word $\times$ Q Bit) $\kappa_{mem} \approx 600$ fW/bit-Hz (6)

I/Q: $P_{I/O} = \kappa_{I/O} Bf_{I/O}$ (watts)
(B Bits) $\kappa_{I/O} \approx 315$ pW/bit-Hz (7)

These PFA values are for static CMOS circuits implemented with a common but specific design style. Other styles (i.e., pseudo-CMOS, different multiplier recoding, etc.) yield different values [11].

Figure 2 shows actual values of multiplier PFA constants, κ_{mult}. These values were determined from (5) using the reported power dissipation from actual multiplier chips of various size. As can be seen from figure 2, the actual values of κ_{mult} are fairly close to the expected value of κ_{mult} = 15 pw/bit^2-Hz. Comparisons for other circuit types can be found in [11].

The power dissipation of arithmetic processing elements depends on what type of operations are being performed, the type of arithmetic used, and the degree of programmability. However, almost all commonly used processing elements are built from integer multipliers, adders, and memory. This includes the common multiply-add based PE and complex butterfly. Also, table look-up, Newton-Raphson iteration, Taylor series expansion, and CORDIC techniques are all composed of integer multipliers, adders, and memory. PEs containing higher order operators based on these techniques such as division and unitary operators ($1/x$, $\log(x)$, x^r, trig functions, etc.) are also composed of these basic functions. The same comments hold for floating point arithmetic which is realized using integer sub-operations. Thus, the power dissipation of almost any arithmetic PE can be computed from the PFA equations for integer addition, multiplication (5), and memory (6)

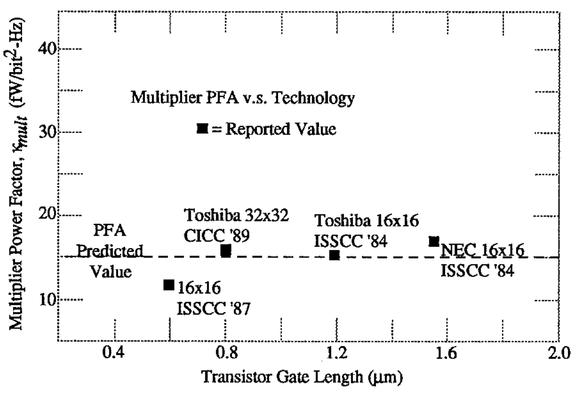


Fig. 2. Multiplier PFA constant values taken from actual chips.

by summing the contribution from each element. The power dissipation of entire VLSI chips composed of arrays of PEs can be computed in a similar manner.

3. Power Dissipation of Localizable Algorithms

Computational problems which can be localized into a regular or semi-regular series of operations are best able to exploit the computational power of VLSI array processors. This is a very broad class of algorithms and includes filtering, most transforms, matrix decompositions, and least squares solutions as applied to spectral estimation, beamforming, speech and image processing. A localizable algorithm can be modeled by an N-dimensional computational flow graph where each index point represents a processing element and dependencies are indicated by arcs connecting the index points. A good deal of research has been published on mapping localizable algorithms into arrays of identical or near-identical processing elements [2]. The flow graph of most useful algorithms is many times larger than the size of an array that can fit on a single VLSI chip. Techniques for partitioning large computational arrays into blocks of identical smaller arrays have been explored by several authors [5]–[6]. These blocks of identical smaller arrays can be implemented by a single VLSI processor array using FIFO registers to provide data communication between the blocks. In this section, a method is presented for estimating the power dissipation of an algorithm as a function of the processor array to which it has been mapped based on the PFA power estimation approach outlined in Section 2. We consider the class of algorithms described by 1, 2, and 3 dimensional computational flow graphs implemented on linear, hex, and cubic VLSI array processors.

For large 1D, 2D, or 3D, computational flowgraphs to be implemented in VLSI chips with a finite number of processing elements, two basic design problems exist: 1) what is the best way to partition the computations onto smaller processor arrays, and, 2) what is the best way to arrange the processing elements; linear, hex, or cubic. Moldovan and Fortes [5] have developed algorithms for partitionment of large computation flowgraphs onto smaller fixed size processing arrays suitable for VLSI implementation. These algorithms form the foundation necessary for computer-aided exploration of partitioning alternatives. However, these mapping and partitioning strategies consider only speed and area for optimization metrics. The closed form equations for estimating power dissipation developed in this section serve as a third optimization metric for computer-aided partitioning of computational algorithms onto VLSI array processors.

3.1. General Considerations

A VLSI array processor performs three main functions: computation, I/O, and inter-block data communication via FIFO registers. The total power dissipation of a VLSI array processor can be divided into these three areas:

$$P = P_{PE} + P_{I/O} + P_{FIFO} \text{ (watts)} \qquad (8)$$

Where:

P_{PE} = Power dissipated by all processing elements

$P_{I/O}$ = Power dissipated inputting and outputting data

P_{FIFO} = Power dissipated by all FIFO registers

The characteristic properties of array processors (high degree of modularity, localized interconnection, absence of central control) permits the power of M PEs to be estimated as M times the power of a single PE; a basic requirement of the PFA power estimation method. As indicated in Section 2, the power dissipation of most commonly used PEs can be calculated using the PFA equations for integer addition, multiplication, and memory. It is useful to divide P_{PE}, into a computational component, P_{comp}, and a memory component, P_{mem}. Note that P_{comp} will be M times the computational power dissipation of a single PE regardless of the partitioning strategy. For example, 20 Q bit multipliers operating at 100 kHz will dissipate approximately the same power at 5 Q bit multipliers operating at 400 kHz since the power has been shown to be proportional to Q^2f. It will be shown, however, that P_{mem} is not independent of partitioning.

We have chosen specific, although representative, configurations to analyze so that we could present closed form equations which indicate the general trends of power dissipation as a function of both algorithm complexity and array processor complexity. In addition, we consider the power dissipation of an array processing system characteristically optimized for a specific algorithm or algorithms within the same class; such as systolic arrays. Systolic arrays are less suitable for implementation of algorithms of widely varying categories, which then would call for a general purpose processor or an array of hetereogeneous processors. Research into the optimized FIFO and local memory structure feeding between systolic arrays sequenced to solve problems requiring composite algorithms is still an open area of research dedicated to the implementation of a specific computational algorithm.

3.2. Linear Processor Array

We first consider the case of an algorithm with a 1 dimensional computational flow graph, such as a linear filter, requiring M processing elements which is to be implemented on a linear processor array with m PEs. As illustrated in figure 3, the algorithm is partitioned into $\lceil M/m \rceil$ blocks ($\lceil \cdot \rceil$ is the next highest integer or ceiling function) and requires only a single word FIFO register. If the M processing elements each contain S words of internal memory to store coefficients or state information (shown explicitly in figure 3), the m PEs of the linear array must each contain $W = \lceil M/m \rceil \times S$ words of internal memory. To obtain an effective throughput of f samp/sec, the m memory blocks must operate at $f_{PE} = \lceil M/m \rceil Sf$ samp/sec (single port memories). Using equation (6), the memory component of P_{PE} is given by:

$$P_{mem} = \kappa_{mem} m \lceil M/m \rceil^2 S^2 Qf \text{ (watts)} \qquad (9)$$

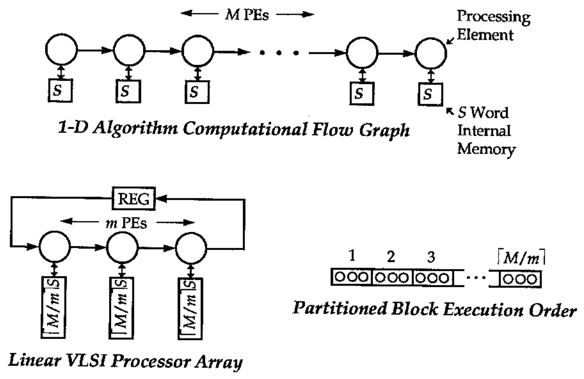
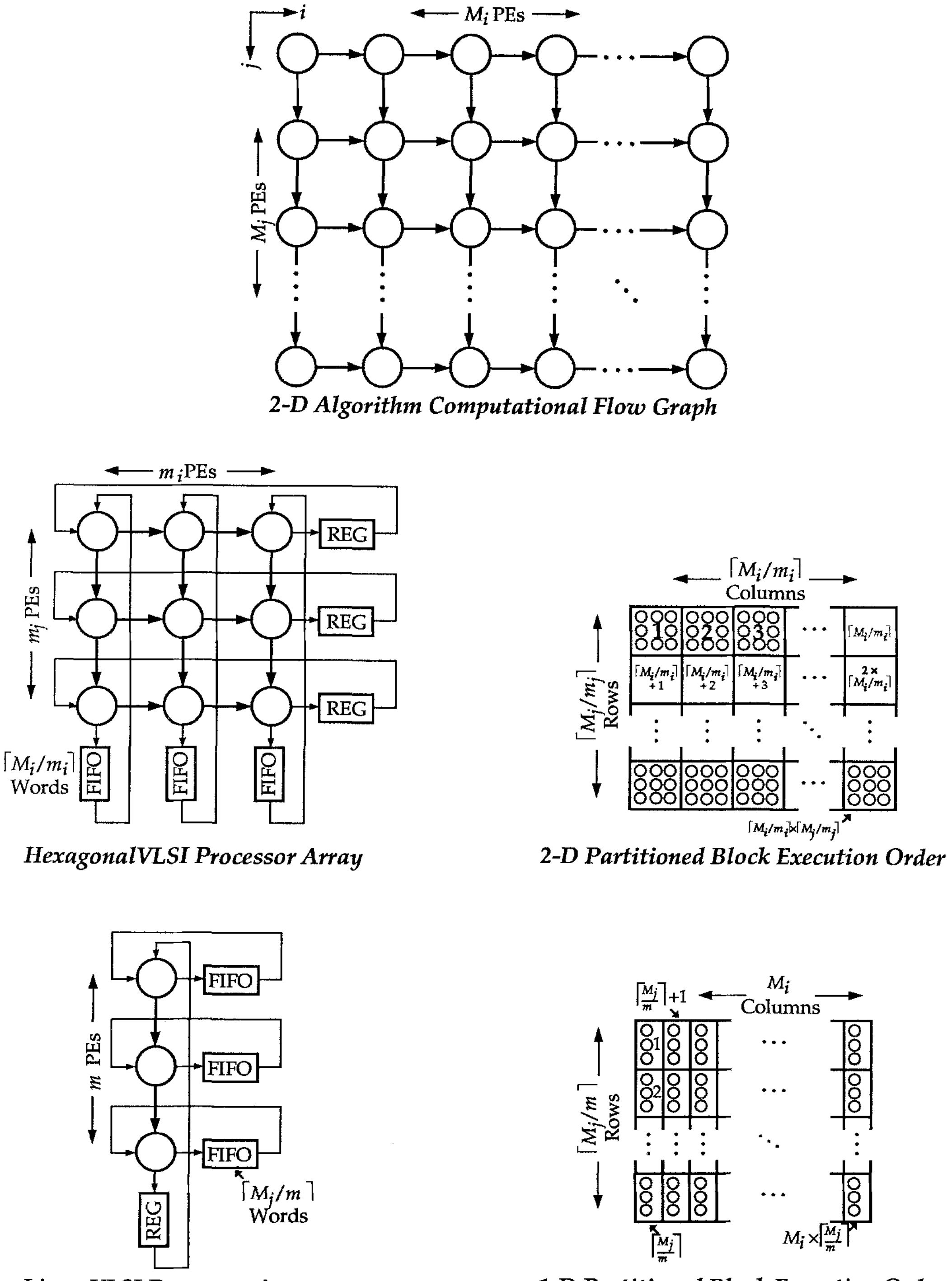

M PEs
Processing Element
S
S Word Internal Memory
1-D Algorithm Computational Flow Graph
REG
m PEs
⌈M/m⌉S
1 2 3 ⌈M/m⌉
Partitioned Block Execution Order
Linear VLSI Processor Array

Fig. 3. One-dimensional algorithm implemented on a linear processor array.

We next consider the case of an algorithm with a 2 dimensional computational flow graph, such as a matrix-vector multiplication, with M_i columns and M_j rows which is to be implemented on a linear processor array with m PEs. Figure 4 illustrates this case with the block ordering that results in the shortest FIFO registers. The same analysis can be applied to other partitioning strategies. To obtain an effective throughput



i
j
M_i PEs
M_j PEs
2-D Algorithm Computational Flow Graph
m_iPEs
m_j PEs
REG
FIFO
$\lceil M_i/m_i \rceil$ Words
HexagonalVLSI Processor Array
$\lceil M_i/m_i \rceil$ Columns
$\lceil M_j/m_j \rceil$ Rows
1
2
3
$\lceil M_i/m_i \rceil$
$\lceil M_i/m_i \rceil$ +1
$\lceil M_i/m_i \rceil$ +2
$\lceil M_i/m_i \rceil$ +3
2 × $\lceil M_i/m_i \rceil$
$\lceil M_i/m_i \rceil \times \lceil M_j/m_j \rceil$
2-D Partitioned Block Execution Order
m PEs
$\lceil M_j/m \rceil$ Words
LinearVLSI Processor Array
$\lceil \frac{M_j}{m} \rceil$+1
M_i Columns
$\lceil M_j/m \rceil$ Rows
$\lceil \frac{M_j}{m} \rceil$
$M_i \times \lceil \frac{M_j}{m} \rceil$
1-D Partitioned Block Execution Order

Fig. 4. Two-dimensional algorithm implemented on a hexagonal and linear processor array.

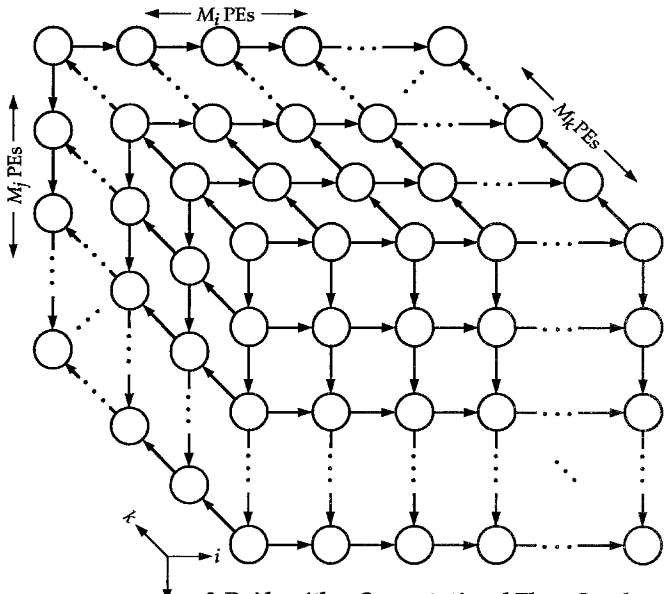
rate of f vector-samp/sec, the linear processor array must operate at $M_i \lceil M_j/m \rceil f$ samp/sec. There will be m FIFO registers of length $\lceil M_j/m \rceil$ in the i direction and a single word FIFO in the j direction. Using (6), the power dissipated by the FIFO registers is given by:

$$P_{FIFO} = \kappa_{mem}(1 + m \lceil M_j/m \rceil)QM_i \lceil M_j/m \rceil f \text{(watts)} \quad (10)$$

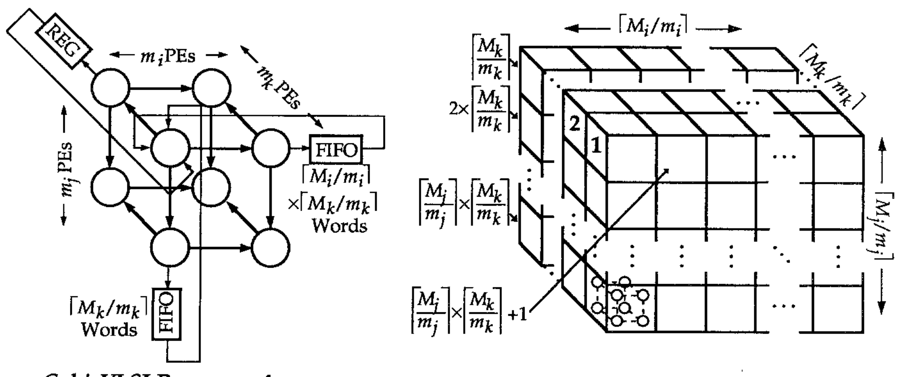
If S words of memory are required at each node of the flow graph (not shown explicitly on figure 4), the m PEs of the linear array must each contain $W = mM_i \lceil M_j/m \rceil \times S$ words of internal memory operating at a rate of $SM_i \lceil M_j/m \rceil f$ samp/sec. The memory component of P_{PE} is given by:

$$P_{mem} = \kappa_{mem}m(M_i \lceil M_j/m \rceil)^2 S^2 Qf \text{(watts)} \quad (11)$$

For the case of a 3-D computational flow graph, such as matrix-matrix multiplication, implemented with a linear processor array, many alternative block orderings are possible. The block ordering shown in figure 5 results in the shortest FIFO registers. To obtain an effective throughput rate of f array-samp/sec, the linear processor array must operate at $M_k M_i \lceil M_j/m \rceil f$ samp/sec. There will be m single word registers in the k direction (not shown), m FIFO registers of length $M_k \lceil M_j/m \rceil$ in the i direction, and an M_k word FIFO in the j direction. Using (6), the power dissipated by the FIFO registers is given by:



3-D Algorithm Computational Flow Graph


REG
m_iPEs
m_kPEs
m_j PEs
FIFO
$\lceil M_i/m_i \rceil$ $\times \lceil M_k/m_k \rceil$ Words
$\lceil M_k/m_k \rceil$ Words
FIFO
$\lceil M_i/m_i \rceil$
$\lceil M_k/m_k \rceil$
$\lceil M_j/m_j \rceil$
$\lceil M_k/m_k \rceil$
$2\times \lceil M_k/m_k \rceil$
$\lceil M_i/m_j \rceil \times \lceil M_k/m_k \rceil$
$\lceil M_i/m_j \rceil \times \lceil M_k/m_k \rceil +1$
2
1

CubicVLSI Processor Array
(FIFOs only shown on 1st PE)

3-D Partitioned Block Execution Order

Fig. 5. (a) Three-dimensional algorithm implemented on a cubic processor array.

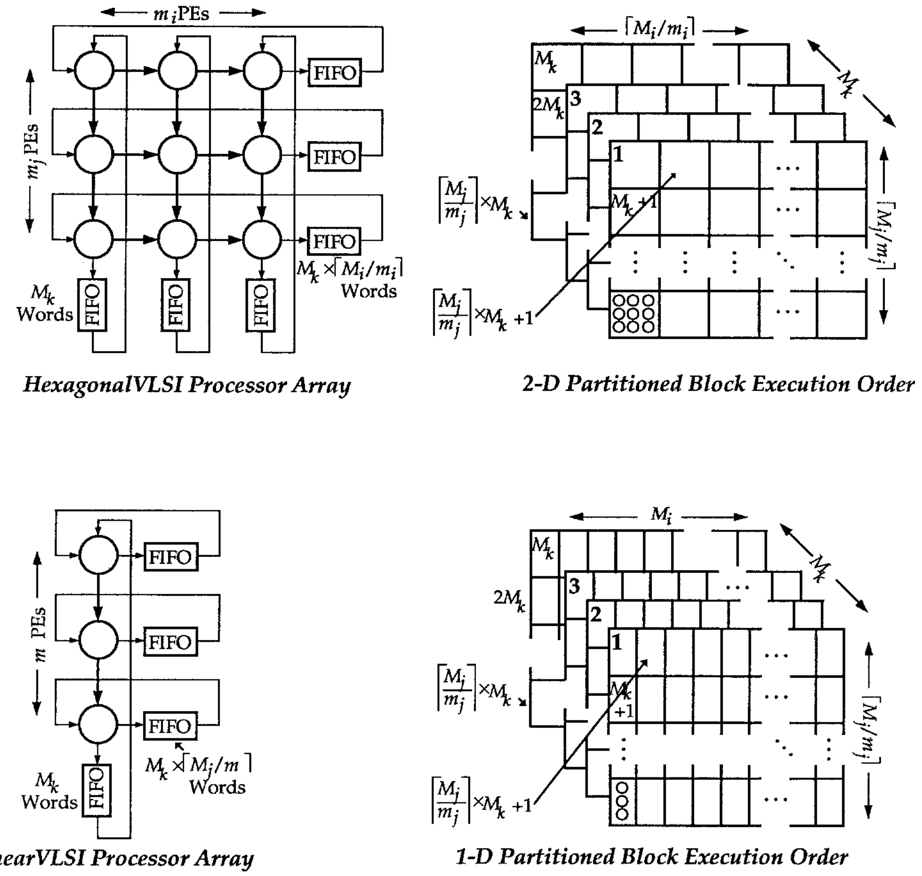

m_iPEs
m_j PEs
FIFO
M_k Words
$M_k \times \lceil M_i/m_i \rceil$ Words
HexagonalVLSI Processor Array
$\lceil M_i/m_i \rceil$
M_k
$2M_k$
3
2
1
M_k+1
$\lceil M_j/m_j \rceil \times M_k$
$\lceil M_j/m_j \rceil \times M_k+1$
$\lceil M_j/m_j \rceil$
2-D Partitioned Block Execution Order
m PEs
$M_k \times \lceil M_j/m \rceil$ Words
LinearVLSI Processor Array
M_i
1-D Partitioned Block Execution Order

Fig. 5. (b) Three-dimensional algorithm implemented on hex and linear processor arrays.

$$P_{FIFO} = \kappa_{mem}(m + M_k + mM_k \lceil M_j/m \rceil)QM_kM_i \lceil M_j/m \rceil f \text{ (watts)} \quad (12)$$

If S words of memory are required at each node of the flow graph, the m PEs of the linear array must each contain $W = mM_kM_i \lceil M_j/m \rceil \times S$ words of internal memory operating at a rate of $SM_kM_i \lceil M_j/m \rceil f$ samp/sec. The memory component of P_{PE} is given by:

$$P_{mem} = \kappa_{mem} m(M_kM_i \lceil M_j/m \rceil)^2 S^2 Qf \text{ (watts)} \quad (13)$$

Using equations (7) and (8)-(13), table 1 presents the overall power dissipation of a linear array processor as a function of the size and dimensionality of the algorithm being implemented. It can be seen that the only components which have a dependence on the dimensionality of the algorithm are P_{FIFO} and $P_{I/O}$. All other components are linearly or quadratically proportional to the total number of nodes in the algorithm's computational flow graph. Likewise, only P_{FIFO} is effected by the dimensionality of the processor array whereas the other components depend only on the total number of PEs in the processor array. Notice also that partitioning causes the P_{mem} portion of the overall PE power dissipation to increase whereas the P_{comp} portion remains constant and is independent of the partitioning strategy. This implies that the overall power dissipation will be monotonically reduced as the size of the linear array is increased up to the point where m is equal to the size of one of the computational graph dimensions; M_j in the cases considered here.

3.3. Hexagonal Processor Array

The case of an algorithm with a 2-D $M_i \times M_j$ computational flow graph being implemented on a 2-D $m_i \times m_j$ hexagonal array processor is shown in figure 4. The analysis is similar to that of the linear array processor. An effective throughput rate of f vector-samp/sec is obtained when the array processor operates at $\lceil M_i/m_i \rceil \times \lceil M_j/M_j \rceil f$ vector-samp/sec. For the block ordering shown in figure 4, there will be m_j single word FIFO registers in the i direction and m_i FIFO registers containing $\lceil M_i/m_i \rceil$ words in the j direction,, Using (6), P_{FIFO} is given by:

Table 1. Power dissipation of 1, 2, and 3 dimensional algorithms implemented on linear, hex, and cubic array processors.

Algorithm Dimension	Target Array	VLSI Array Processor Power Dissipation (Watts)
1-D	Linear	$Mp_{comp} + \kappa_{mem} m \lceil M/m \rceil^2 S^2 Qf + \kappa_{I/O} Bf$
2-D	Linear	$Mp_{comp} + \kappa_{mem} m(M_i \lceil M_j/m \rceil)^2 S^2 Qf + \kappa_{I/O} Bf + \kappa_{mem}(1 + m \lceil M_j/m \rceil) QM_i \lceil M_j/m \rceil f$
3-D	Linear	$Mp_{comp} + \kappa_{mem} m(M_k M_i \lceil M_j/m \rceil)^2 S^2 Qf + \kappa_{I/O} Bf + \kappa_{mem}(m + M_k + mM_k \lceil M_j/m \rceil) QM_k M_i \lceil M_j/m \rceil f$
2-D	Hex	$Mp_{comp} + \kappa_{mem} m_i m_j \lceil M_i/m_i \rceil^2 \lceil M_j/m_j \rceil^2 S^2 Qf + \kappa_{I/O} Bf + \kappa_{mem}(m_j + m_i \lceil M_i/m_i \rceil) Q \lceil M_i/m_i \rceil \lceil M_j/m_j \rceil f$
3-D	Hex	$Mp_{comp} + \kappa_{mem} m_i m_j M_k^2 \lceil M_i/m_i \rceil^2 \lceil M_j/m_j \rceil^2 S^2 Qf + \kappa_{I/O} Bf + \kappa_{mem}(m_i m_j + m_i M_k + m_j M_k \lceil M_i/m_i \rceil) \times QM_k \lceil M_i/m_i \rceil \lceil M_j/m_j \rceil f$
3-D	Cubic	$Mp_{comp} + \kappa_{mem} m_i m_j m_k \lceil M_i/m_i \rceil^2 \lceil M_j/m_j \rceil^2 \lceil M_k/m_k \rceil^2 S^2 Qf + \kappa_{I/O} Bf + \kappa_{mem} Q \lceil M_i/m_i \rceil \lceil M_j/m_j \rceil \lceil M_k/m_k \rceil f \times (m_i m_j + m_i m_k \lceil M_k/m_k \rceil + m_j m_k \lceil M_i/m_i \rceil \lceil M_k/m_k \rceil)$

Where: M = Total number of flow graph nodes
M_n = Number of flow graph nodes along the n^{th} index
m_n = Number of PEs along the n^{th} processor array index
p_{comp} = Power dissipated by a single PE
f = effective throughput rate
S = internal storage per flow graph node (words)
Q = Wordlength
B = Number of I/O bits
$\lceil \cdot \rceil$ = The next highest integer or ceiling function

$$P_{FIFO} = \kappa_{mem}(m_j + m_i \lceil M_i/m_i \rceil) Q \lceil M_i/m_i \rceil M_j/m_j \rceil f \quad \text{(watts)} \quad (14)$$

If the PEs in the computational flow graph contain S words of internal memory, each PE in the hex array processor must contain $\lceil M_i/m_i \rceil \lceil M_j/m_j \rceil S$ words of memory. The memory portion of P_{PE} is given by:

$$P_{mem} = \kappa_{mem} m_i m_j \lceil M_i/m_i \rceil^2 \lceil M_j/m_j \rceil^2 S_2 QF \text{ (watts)} \quad (15)$$

Figure 5 shows one possible block ordering for implementing a 3-D computational flow graph with a 2-D processor array. In this case, an effective throughput rate of f array-samp/sec is obtained when the array processor operates at $M_k \lceil M_i/m_i \rceil \times \lceil M_j/m_j \rceil f$ vector-samp/sec. There will be $m_i m_j$ single word registers in the k direction (not shown), m_i FIFO registers each containing M_k words the j direction, and m_j FIFO registers each containing $M_k \lceil M_i/m_i \rceil$ words in the i direction. Using (6), P_{FIFO} is given by:

$$P_{FIFO} = \kappa_{mem}(m_i m_j + m_i M_k + m_j M_k \lceil M_i/m_i \rceil) QM_k \lceil M_i/m_i \rceil \lceil M_j/m_j \rceil f \quad (16)$$

If the PEs in the computational flow graph contain S words of internal memory, each PE in the hex array processor must contain $M_k \lceil M_i/m_I \rceil \lceil M_j/m_j \rceil S$ words of memory. The memory portion of P_{PE} is given by:

$$P_{mem} = \kappa_{mem} m_i m_j M_k^2 \lceil M_i/m_i \rceil^2 \lceil M_j/m_j \rceil^2 S^2 Qf \text{ (watts)} \quad (17)$$

Using equations (7), (8), and (14)-(17), table 1 presents the overall power dissipation of a hexagonal array processor as a function of the size and dimensionality of the algorithm being implemented. As with the linear array, the only power dissipation components which have a dependence on the dimensionality of the algorithm are P_{FIFO} and $P_{I/O}$ and only the P_{FIFO} component depends on the dimensionality of the processor array. It can also be observed that the overall power dissipation is reduced as the size of the processor array is increased up to the point where one or both of the processor array dimensions is equal to the computational flow graph dimensions; $m_j = M_j$ and $m_i = M_i$ in the cases considered here.

3.4. Cubic Processor Array

In this section, we extend our approach to a hypothetical three dimensional processor array. Care must be taken in applying these results since the implementation of a processor array with more than two dimensions on a planer layout may violate the assumption of localized interconnection. Although current VLSI technologies permit only planer layouts, some work is being done in the areas of massively interconnected flip-chip technology and bonded-silicon to permit a third dimension.

Figure 5 illustrates a 3-D computational flow graph partitioned using a cubic processor array. Each PE must have FIFO connections in three directions. Only the FIFOs for the first PE are shown for clarity. The cubic array processor must operate at $\lceil M_i/m_i \rceil \times \lceil M_j/m_j \rceil \times \lceil M_k/m_k \rceil f$ array-samp/sec to sustain an effective throughput rate of f array-samp/sec. For the block ordering shown in figure 5, there will be $m_i m_j$ single word FIFOs in the k direction, $m_i m_k$ FIFOs in the j direction of $\lceil M_k/m_k \rceil$ words each, and $m_j m_k$ FIFOs in the i direction of $\lceil M_i/m_i \rceil \lceil M_k/m_k \rceil$ words each. Using (6), P_{FIFO} is given by:

$$P_{FIFO} = \kappa_{mem}(m_i m_j + m_i m_k \lceil M_k/m_k \rceil + m_j m_k \lceil M_i/m_i \rceil \lceil M_k/m_k \rceil) \times Q \lceil M_i/m_i \rceil \lceil M_j/m_j \rceil \lceil M_k/m_k \rceil f \text{ (watts)} \tag{18}$$

If the PEs in the computational flow graph contain S words of internal memory, each PE in the cubic array processor must contain $\lceil M_i/m_i \rceil \lceil M_j/m_j \rceil \lceil M_k/m_k \rceil S$ words of memory. The memory portion of P_{PE} is given by:

$$P_{mem} = \kappa_{mem} m_i m_j m_k \lceil M_i/m_i \rceil^2 \lceil M_j/m_j \rceil^2 \lceil M_k m_k \rceil^2 S^2 Qf \text{ (watts)} \tag{19}$$

The resulting total power dissipation of a cubic array processor as a function of the size of the 3-D algorithm being implemented is presented in table 1. As with the 1-D and 2-D array processors, the overall power dissipation is reduced as the size of the cubic array processor is increased until a dimension of the processor array is equal to a dimension of the computational flow graph.

4. Application to 15 × 15 Matrix Multiply

As an example application of the power dissipation analyses presented in earlier sections, consider the single chip VLSI implementation of an integer multiplication between two 15 by 15 matrices. Figure 6 shows the computational flow graph for this operation based on

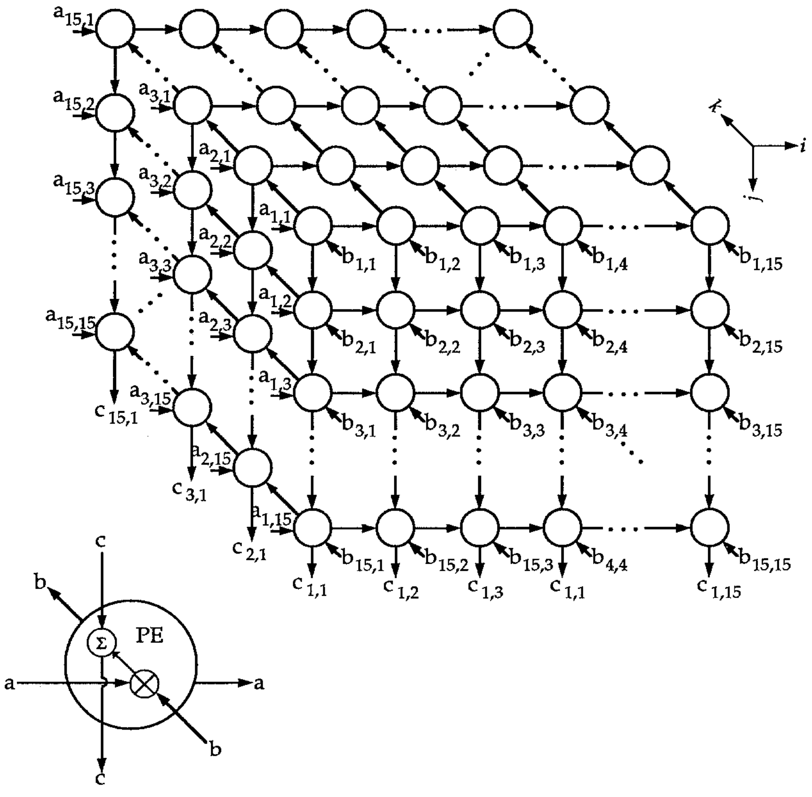


Fig. 6. Computational flow graph for a 15 × 15 matrix multiplication.

processing elements which contain a single multiplier and a single adder. Assume that 16 bit words are to be used and that we are constrained by silicon area to using at most 8 multiply-add processing elements. The 8 PEs can be configured as a linear, hex, or cubic connected processing array. Assume also a maximum PE operating rate of 40 MHz; a reasonable value for near 1 μm CMOS technologies. Since there will be approximately 500 partitioned blocks ($15 \times 15 \times 15/8$ + boundaries), 15×15 output matrices can be calculated at a maximum rate of 80 kHz. In this section, we determine the power dissipation of the matrix multiply for each processor array alternative. We use the block partitioning order shown in figures 2–4 so that the equations of table 1 can be used; although similar equations could be developed for other partitioning strategies. The design problem is summarized in table 2.

Table 2. Design problem for power dissipation example.

Operation:	15×15 Matrix Multiplication
Type of Arithmetic:	16 bit integer
Type of PE:	Multiply-add
Number of Available PEs:	8
Sample Rate:	80 kHz (array-outputs/sec)

Values for the PFA constants, κ, are taken as those given in equations (5)–(7). Values for the parameters used in the power dissipation equations of table 1 which are independent of the configuration of the processor array are: $M = 3375$, $M_i = M_j = M_k = 15$, and $Q = 16$ bits. The total number of output bits in the output matrix will be: $B = 15 \times 15 \times Q = 3600$. Since there is no memory required internal to the PEs, the parameter S is taken to be zero.

A linear processor array and a block partitioning order is shown in figure 7. Notice that the size of the linear array is not matched to the size of the problem causing the last PE (PE8) to remain idle during the lower block operations (16–30, 46–60, . . .). Notice also that registers are not necessary in the k direction since the $b_{r,c}$ elements are transmittent and can simply be held during the evaluation of a whole column. There are 450 total blocks which must be performed. The first 7 PEs must operate in all 450 blocks at an effective rate of $f = 457f_s$ (36 MHz) whereas the last PE need only operate in 225 blocks at an effective rate of $f = 225f_s$. Assuming the PE power dissipation to be dominated by the multiplier and using equation (5), the overall PE power dissipation is:

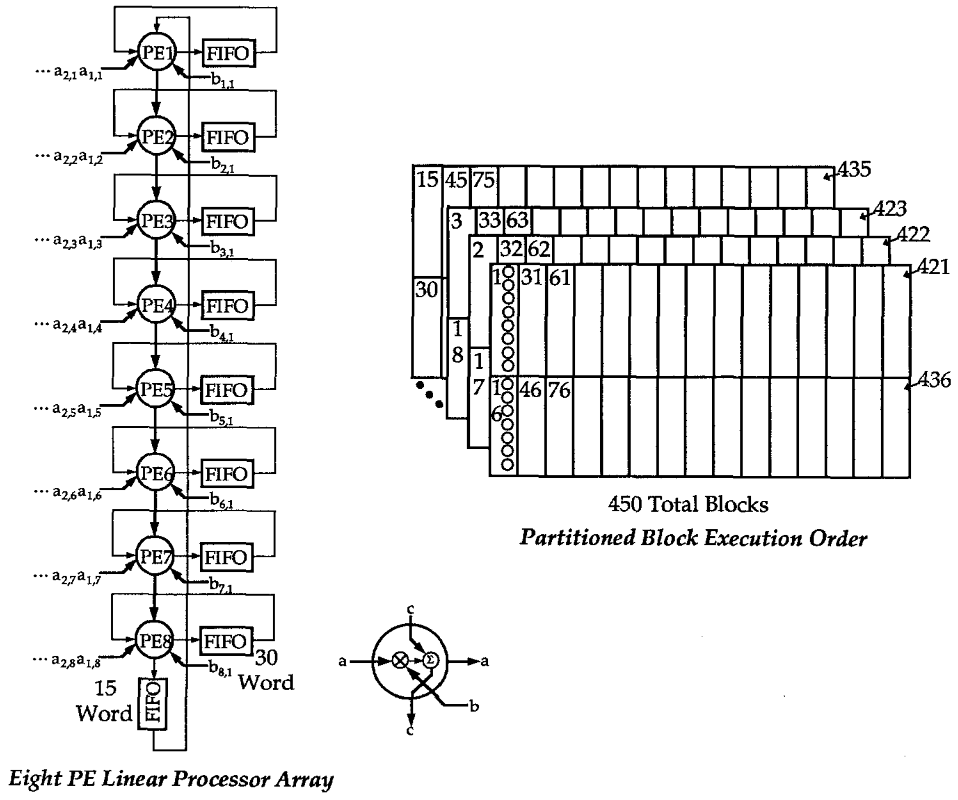


Fig. 7. Linear processor array and block partitioning order for 15×15 matrix multiplication example.

$$\begin{aligned} P_{PE} &= 7 \times \kappa_{mult} Q^2 450 f_s + \kappa_{mult} Q^2 225 f_s \\ &= 3375\{\kappa_{mult} Q^2 f_s\} \\ &= M p_{PE} \end{aligned} \tag{20}$$

Where M is the total number of flow graph nodes $= 15 \times 15 \times 15 = 3375$.

Thus, as stated earlier and shown in the equations of table 1, the computational component of the overall PE power dissipation is equal to the total number of flow graph nodes times the power dissipated by a single PE and is independent of the partitioning strategy. Assuming that the power dissipation of a single PE is dominated by the multiplier, equation (5) can be used to predict p_{PE}. Table 3 contains the major components and the overall power dissipation of the linear array processor of figure 7 based on the third equation of table 1 with $m = 8$.

Table 3. Matrix multiplication example using 16 bit integer multipliers: negligible power difference.

Target Array	P_{PE} (mW)	$P_{I/O}$ (mW)	P_{FIFO} (mW)	P_{TOTAL} (mW)
Linear	1100	91	83	1274
Hex	1100	91	110	1301
Cubic	1100	91	115	1306

A hexagonal processor array and a block partitioning order is shown in figure 8. As with the linear array, registers are not necessary in the k direction because of the transmittent b variables. There are 480 total blocks which must be performed increasing the PE operating rate from the linear array's 36 MHz to 480 $\times$ 80 kHz = 38.4 MHz. Note that the lower row of PEs (PE5–PE8) are idle on the bottom-most partitionment blocks and the right column of PEs (PE4 and PE8) are idle on the right-most blocks. Even though the frequency has been increased, the PE utilization has been decreased causing the PE power (P_{PE}) to remain the same. As with the linear array, the overall PE power dissipation can be shown to be given as the product of the total number of nodes in the original flow graph and the power dissipated by a single PE. The overall power dissipation of the hexagonal processor array is given by the fifth equation of table 1 with $m_i = 4$ and $m_j = 2$. The major components and the overall power dissipation of the hexagonal array processor of figure 8 are shown in table 3.

A cubic processor array and block partitioning order is shown in figure 9. Only the FIFOs for the first PE are shown in the figure for clarity (k direction registers not necessary). There are 512 total blocks which must be performed requiring the PEs to operate at a top rate of about 41 MHz. As mentioned earlier, cubic processor arrays implemented as a planer VLSI layout may violate the assumptions of localized interconnection causing the equation of table 1 to predict an overly optimistic power dissipation. The overall power dissipation of the cubic processor array is given by the last equation of table 1 with $m_i = 2$, $m_j = 2$, and $m_k = 2$. The major components and the overall power dissipation of the cubic array processor of figure 9 are shown in table 3.

For the 15 $\times$ 15 matrix multiplication example implemented with 8 PEs, the dimensionality of the processor array has little effect on the overall power dissipa-

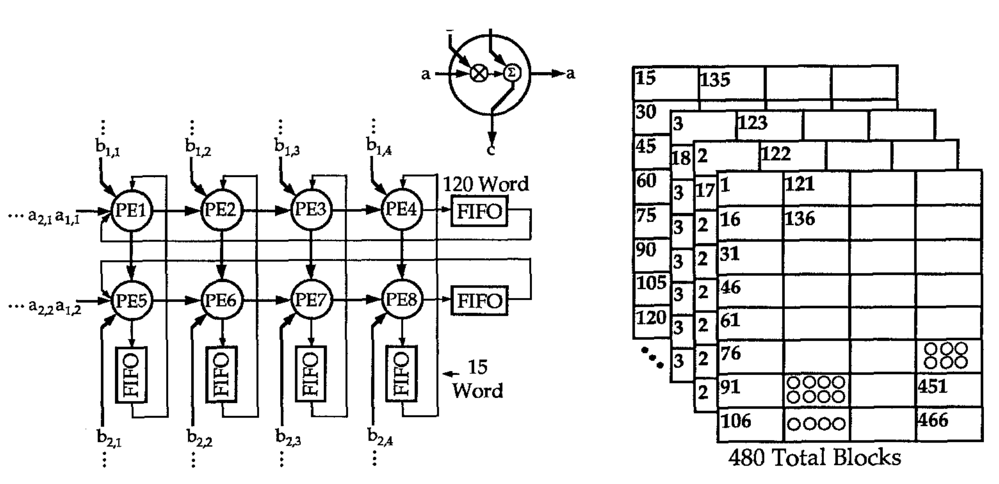
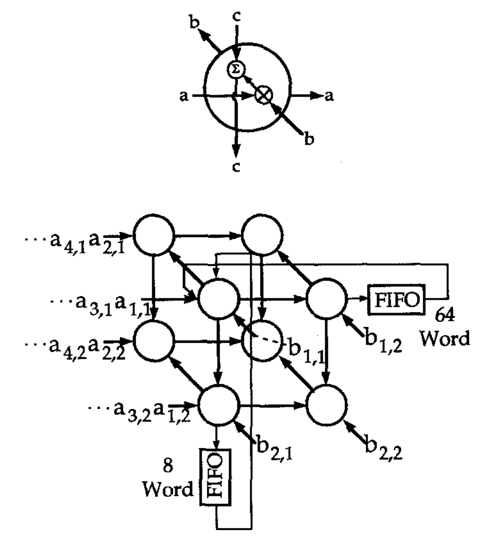
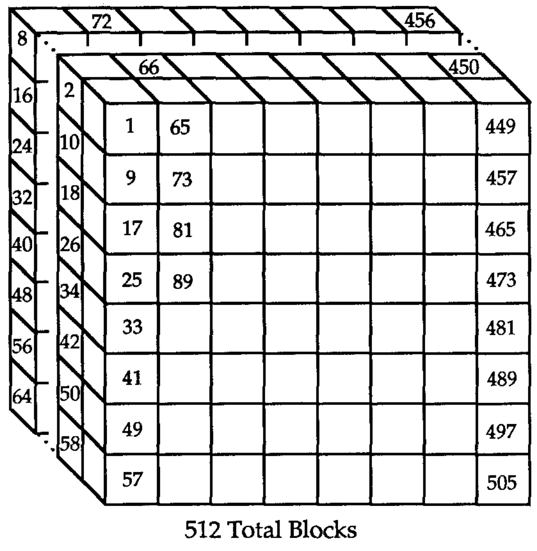


Fig. 8. Hexagonal processor array and block partitioning order for 15 $\times$ 15 matrix multiplication example.



8 PE CubicVLSI Processor Array
(FIFOs only shown on 1st PE)



Partitioned Block Execution Order

Fig. 9. Cubic processor array and block partitioning order for 15 × 15 matrix multiplication example.

tion. However, consider the case where the elements of one of the matrices are constrained to be a power of two. In this case, the PE is reduced to a simple shift and add operation with:

$$p_{comp} = \kappa_{add}Qf \text{ (watts)} \tag{21}$$

giving

$$p_{comp} = Mp_{PE} \approx 60 \text{ mW} \tag{22}$$

Further, consider the case where the result of the matrix multiply is to be added to a third matrix of fixed but arbitrary constants. In this case, the PE must contain a storage element giving the parameter S in the equations of table 1 a value of one. Under these circumstances, the major components and overall power dissipation of the three alternative processor array configurations are given in table 4.

Table 4. Modified matrix multiply example using 2^n multipliers: significant power difference.

Target Array	P_{PE} (mW)	$P_{I/O}$ (mW)	P_{FIFO} (mW)	P_{TOTAL} (mW)	Percent Increase
Linear	1313	91	83	1487	0%
Hex	1485	91	110	1686	13%
Cubic	1680	91	115	1886	27%

These results indicate that overall power dissipation tends to increase with the dimensionality of a processor array if the size of a processor array is not well matched to the size of an intended algorithm's computational flow graph. The higher the dimensionality of the target processor array, the more coarsely grained the block partitioning resulting in an increase in the required PE processing rate and a rise in the overall power dissipation.

5. Conclusion

We first reviewed the power factor approximation (PFA) power estimation technique leading to power dissipation equations for memory, I/O, and multiplication. The PFA approach was extended to estimate the power dissipation of an algorithm as a function of the processor array to which it is mapped. We presented closed form power dissipation equations for the class of algorithms described by 1, 2, and 3 dimensional computational flow graphs implemented on linear, hex, and cubic processor arrays. A 15 × 15 matrix multiplication example was used to demonstrate that although the dimensionality of the processor array has little effect on the power dissipated performing computations, the memory power dissipation is a function of the partitioning strategy. This is most pronounced when the size of the target processor array is not well matched to the size of the algorithm. In all cases, the overall power dissipation is reduced as the size of the processor array is increased until a dimension of the processor array is equal to the corresponding dimension of the computational flow graph.

A key application area of the power estimation methods developed in this paper is VLSI algorithm/architecture design, selection, and partitionment. When combined with previously developed area-time metrics, the proposed power dissipation estimation methods permit a more realistic evaluation of new and existing VLSI solutions to computational tasks.

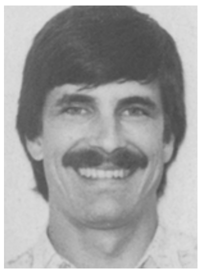
References

1. H.T. Kung, "Why systolic architectures?," *Computer*, 1982, pp. 37–46.
2. S.Y. Kung, *VLSI Array Processors*, Englewood Cliffs, NJ: Prentice-Hall, 1988.
3. William A. Porter, "Concurrent forms of signal processing algorithms," *IEEE Trans. on Circuits and Systems*, vol. 36, 1989, pp. 553-560.
4. Clark D. Thompson, "Area-time complexity for VLSI," *Proc. of 3rd International Conference on Artificial Intelligence and Information-Control Systems of Robots*, 1984, pp. 373-382.
5. Dan I. Moldovan and Jose A.B. Fortes, "Partitioning and mapping algorithms into fixed size systolic arrays," *IEEE Trans. on Computers*, vol. c-35, 1986, pp. 1-12.
6. Kai Hwang and Yeng-Heng Cheng, "Partitioned matrix algorithms for VLSI arithmetic systems," *IEEE Trans. on Computers*, vol. c-31, 1982, pp. 1215-1224.
7. Guo-Jie Li and Benjamin W. Wah, "The design of optimal systolic arrays," *IEEE Trans. on Computers*, vol. c-34, 1985, pp. 66-77.
8. Scott R. Powell and Paul M. Chau, "A model for estimating power dissipation in a class of DSP VLSI chips," *IEEE Transactions on Circuits and Systems*, vol. 38, 1991, pp. 646-649.
9. J.S. Ward, et al., "Figures of merit for VLSI implementations of digital signal processing algorithms," *IEE Proceedings*, vol. 131, Part F, 1984, pp. 64-70.
10. Clark K. Thompson and Prabhakar Raghavan, "On estimating the performance of VLSI circuits," *Proc. Conference on Advanced Research in VLSI*, 1984, pp. 34-44.
11. Scott R. Powell and Paul M. Chau, *VLSI Signal Processing IV*, IEEE Press, 1990.
12. David A. Hodges and Horace G. Jackson, *Analysis and Design of Digital Integrated Circuits*, New York: McGraw-Hill, 1983.
13. Ghassan Y. Yacoub and Walter H. Ku, "An accurate simulation technique for short-circuit power dissipation based on current component isolation,," *Proc. IEEE Int. Symp. on Circuits and Systems* (ISCAS), 1989, pp. 1157-1161.
14. Harry J.M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. sc-19, 1984, pp. 468-473.
15. Sung Mo Kang, "Accurate simulation of power dissipation in VLSI circuits," *IEEE Journal of Solid-State Circuits*, vol. sc-21, 1986, pp. 889-891.
16. Robert Tjarnstrom, "Power dissipation estimate by switch level simulation," *Proc. IEEE Int. Symp. on Circuits and Systems* (ISCAS), 1989, pp. 881-884.
17. Mehmet A. Cirit, "Estimating dynamic power consumption of CMOS circuits," *Proc. IEEE Int. Conf. on Computer Aided Design* (ICCAD), 1987, pp. 534-537.
18. Akhilesh Tyagi, "Hurcules: A power analyzer for MOS VLSI circuits," *Proc. IEEE Int. Conf. on Computer Aided Design* (ICCAD), 1987, pp. 530-533.

Paul M. Chau received the B.S. degree in Electrical Engineering from the Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, in 1977, where he received the Moore School A. Atwater Kent Prize and the Cornelius N. Weygandt Memorial Award. He received the M.S. degree from Syracuse University, Syracuse, NY in 1981, and the Ph.D. degree from Cornell University, Ithaca, NY in 1987. Dr. Chau is a member of Tau Beta Pi and Eta Kappa Nu.

From 1977 to 1982, he was with the General Electric Company, Syracuse, NY, and worked on radar and sonar systems, computer-aided design, test engineering, and advanced development engineering. He also worked at GE Corporate Research and Development Center on a VLSI floating-point processor project.

Dr. Chau's research interests include VLSI architectures and design for digital signal processing, computer architectures, and floating-point processor systems. He has been a member of the faculty in the Electrical and Computer Engineering department at the University of California at San Diego since 1987.



Scott R. Powell received the B.S. and M.S. degrees in electrical engineering from Oregon State University, Corvallis, Oregon, in 1982 and 1983, respectively. He is currently working toward the Ph.D. degree in electrical engineering as a Hughes Aircraft fellow at the University of California at San Diego.

With over eight years experience in VLSI circuit design and signal processing, he is currently Head of the Analog and Digital Design Section at Hughes Aircraft Company's Technology Center in Carlsbad, CA responsible for research and development of high performance signal processing algorithms, architectures, and integrated circuits. His research interests include reduced complexity signal processing algorithms, sigma-delta data conversion, and VLSI circuit design.