View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Washington University St. Louis: Open Scholarship

Washington University in St. Louis

Washington University Open Scholarship

All Computer Science and Engineering Research

Computer Science and Engineering

Report Number: WUCS-83-2

1983-03-01

Regular Array Processors: Asynchronous Versus Clocked Control

Mark A. Franklin, Donald F. Wann, and Sanjay Dhar

Parallel computing structures consisting of large numbers of processors require synchronization so that data communication among processors is possible. Two basic methods of data synchronization, synchronous and asynchronous, are considered. The synchronous or clocked method uses a global clock for synchronization. Clock skew and clock line charge and discharge times both increase with system size. This decreases the data rates achievable and prevents the design of highly modular systems. The asynchronous method has no global control structure and results in a modular and expandable system with the data rate being independent of system size. It is however pin intensive.... **Read complete abstract on page 2.**

Follow this and additional works at: https://openscholarship.wustl.edu/cse_research

Recommended Citation

Franklin, Mark A.; Wann, Donald F.; and Dhar, Sanjay, "Regular Array Processors: Asynchronous Versus Clocked Control" Report Number: WUCS-83-2 (1983). *All Computer Science and Engineering Research.*
https://openscholarship.wustl.edu/cse_research/855

Department of Computer Science & Engineering - Washington University in St. Louis
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160.

This technical report is available at Washington University Open Scholarship: https://openscholarship.wustl.edu/cse_research/855

Regular Array Processors: Asynchronous Versus Clocked Control

Mark A. Franklin, Donald F. Wann, and Sanjay Dhar

Complete Abstract:

Parallel computing structures consisting of large numbers of processors require synchronization so that data communication among processors is possible. Two basic methods of data synchronization, synchronous and asynchronous, are considered. The synchronous or clocked method uses a global clock for synchronization. Clock skew and clock line charge and discharge times both increase with system size. This decreases the data rates achievable and prevents the design of highly modular systems. The asynchronous method has no global control structure and results in a modular and expandable system with the data rate being independent of system size. It is however pin intensive. These two types of control schemes are modelled and the data rates achievable are determined and compared.

REGULAR ARRAY PROCESSORS:

ASYNCHRONOUS VERSUS CLOCKED CONTROL

M.A. Franklin, D.F. Wann and S. Dhar

WUCS-83-2

Prodeedings 1983
Conference on Information Sciences and Systems
The Johns Hopkins University
March 23-25, 1983

Department of Computer Science
and
Department of Electrical Engineering
Washington University
St. Louis, Missouri 63130

This work was supported in part by NSF Grant MCS-78-20731 and
ONR Contract N00014-80-C-0761.

Proc. 1983 Conf.on Information Sciences and Systems, The Johns Hopkins University

Regular Array Processors: Asynchronous versus Clocked Control

Mark A. Franklin, Donald F. Wann and Sanjay Dhar
Department of Electrical Engineering
Washington University, St. Louis, MO. 63130

ABSTRACT

Parallel computing structures consisting of large numbers of processors require synchronization so that data communication among processors is possible. Two basic methods of data synchronization, synchronous and asynchronous, are considered. The synchronous or clocked method uses a global clock for synchronization. Clock skew and clock line charge and discharge times both increase with system size. This decreases the data rates achievable and prevents the design of highly modular systems. The asynchronous method has no global control structure and results in a modular and expandable system with the data rate being independent of system size. It is however pin intensive. These two types of control schemes are modelled and the data rates achievable are determined and compared.

1.0 INTRODUCTION

The advent of VLSI has led to renewed interest in processing structures which have geometrically regular features and planar topologies. Such structures, referred to as regular array processors, are well suited to VLSI implementation because they have few basic modules and can be laid out easily on silicon. Examples are the systolic array [KUNG80], the wave front processor [KUNG82A], and the CHiP machine [SNYD82]. Regular processor arrays have the desirable properties of being modular, can be pipelined, and have the potential for high speed concurrrent processing.

There are two principal methods which can be used in controlling data movement in such structures. These are referred to as asynchronous (or self-timed) and synchronous (or clocked) control schemes and while they are well know as general and often opposing methodologies, there have been few cases where a quantitative comparison has been made between them on the basis of a commmon system design problem [WANN83].

Although there are numerous papers describing the applications of regular structures, particularly in the signal processing field [CAPP81, YEN81, KUNG82B], most of this literature is concerned with the presentation of algorithms and there is only a limited discussion of how such structures can be controlled and on how one determines their performance in terms of bandwidth or data rate. This paper presents a comparison of the asynchronous and clocked control schemes for a particular linear structure and, although not pursued here, the analysis presented extends to other topologies as well. Included in the analysis is a detailed specification of control procedures using both Petri net diagrams and timing precedence relationships.

In practice, clocked digital designs have usually been preferred due to their relative simplicity and generally lower hardware costs. When systems become physically large however, when their size cannot be predicted in advance, or when there are numerous system inputs which operate independently (and on separate clocks), then the advantages of asynchronous design begin to mount. One example of this is in processor bus design where asynchronous control schemes are common [DIGI81, SUTH79]. Another is in modular computer design where expandability and arbitrary system restructuring are key system features [CLAR67]. In such modular systems determining the appropriate clock period and estimating clock skew is difficult, if not impossible, since the final size and configuration of the system is not known in advance. Designing for a maximum size system, on the other hand, would require such large clock periods that system speed would be inordinately slow. A similar type of problem arises in the VLSI domain when designing structures which are robust over a range of features sizes [SEIT79]. The use of asynchronous control schemes in such environments is a natural solution to designing for system growth (or shrinkage). Design of such control schemes are however, difficult and in general engineers have shunned this approach focusing instead on extending conventional clocked schemes. An interesting tradeoff can be seen here. On the one hand with clocked schemes the design of the control logic is simple while the clock distribution problem is difficult. On the other hand with asynchronous schemes the design of the control logic is difficult while there is no clock distribution problem with which to contend.

All of this relates directly to the control problem encountered with regular array processors. The ideal network should be modular and expandable so that it can be readily used to support a wide range of computations. This points to the use of an asynchronous scheme. But, asynchronous systems are difficult to design. Thus there is a substantial benefit to be obtained if simple models can be developed to predict performance of asynchronous and synchronous controlled arrays prior to detailed design. In this paper we present an analysis aimed at determining accurate delay based models for both types of control strategies.

---

*This work was supported in part by NSF Grant MCS-78-20731 and ONR Contract N00014-80-C-0761.

2.0 CONTROL TECHNIQUES FOR REGULAR ARRAYS

Each of the two types of control schemes will now be examined and data rates obtained as a function of computational delay, intermodule path delay, and clock line delay and skew.

2.1 Synchronous Control

Figure 1 illustrates a one dimensional array. Each module in this array is configured to compute the relation

$$Yj = Yk + A*Xi \qquad (1)$$

where Yj, Yk, and Xi are nx1 column matrices and A is a square nxn matrix. A register/processor model for module j of Figure 1 is shown in Figure 2. Notice that there are five registers and one processor, which consists of a multiplier and an adder. Module j accepts Xi from module i on its left, accepts A from the top, and Yk from module k on its right. Module j then computes a new Y, (Yj), according to (1) and sends it to module i. Module j also takes Xi, and passes it unchanged to module k as Xj. The synchronization of these steps can be accomplished by using a central clock in which standard two-phase techniques are employed. A typical clock waveform is shown in the top part of Figure 4. The two phases are identifed as $\phi 1$ and $\phi 2$. The asserted time for phase-one is $\phi 1$ and for phase-two is $\phi 2$. The interphase times are denoted as $\phi 12$ and $\phi 21$.

One array control procedure is to capture A, Xi, and Yk on the assertion of phase one and then during $\phi 1$, to make the computation of Yj. The length of $\phi 1$ would be adjusted so that this computation would be completed before the end of $\phi 1$. Then, on the assertion of phase two, Xj and Yj would be transferred to modules k and i. This is the control procedure depicted in Figure 2 and a model illustrating the delays in the clock lines is illustrated in Figure 3. In addition to the eight clock delays shown in this figure, there are three other delays that are important. Two of these are intermodule delays associated with the transfer of Yk from module k to module j (dYkj), and the transfer of Xi from module i to module j (dXij). The third delay is associated with the delay in obtaining the coefficients from A and transferring them to module j (dAj). These three delays will be called intermodule delays.

In addition to these constraints, there are constraints imposed by the time necessary to charge and discharge the clock line. For a large array this time can be appreciable. We now develop equations for the intermodule and the clock line constraints.

2.1.1 Intermodule Constraints

The time necessary for module j to complete its processing task is constrained by its interactions with module i, module k and the coefficients contained in A. Equations for these three constraints can be obtained by constructing three timing diagrams corresponding to the transfer of:

(1) Yk to module j
(2) Xi to module j
(3) A to module j

A timing diagram for the first of these three cases is depicted in Figure 4 which shows the timing for transferring Yk to module j. Assume that Xi, Yk and A are available and stable when the phase-one clock is asserted. We can determine the first constraint on the value of the clock period, T, by tracing a path from one assertion of the phase-one clock through the various signal paths that include both processsing and propagation delays, and concluding on the next assertion of the phase-one clock. This path is depicted in Figure 4 and yields the following relation:

$$T1 > \phi 1 + \phi 12 + dYkj + (dc2k - dc1j) \qquad (2)$$

This relation is found by starting at the assertion of phase-one of the of the master clock, proceeding through the delay $\phi 1$ (the time necessary to capture A, Xi, and Yk and to perform the computation of the new Yj), proceeding through the interphase delay $\phi 12$, then through the delay between the assertion of phase-two of the master clock and the assertion of phase-two of the clock at module k (equal to dc2k), then through the delay necessary to transfer the new Yk to module j (this is related to the assertion of the master phase-two clock by the relation dYkj-dc1j).

In a similar fashion the other two constraints on the clock period that are associated with the transfer of Xi to module j (T2), and with the transfer of A to module j (T3), are given below.

$$T2 > \phi 1 + \phi 12 + dXij + (dc2i - dc2j) \qquad (3)$$

$$T3 > \phi 1 + \phi 12 + dAj + (dc2A - dc1j) \qquad (4)$$

Rewriting T1 in a slightly different form we obtain:

$$T1 > \phi 1 + \phi 12 + dYkj + (dc1k - dc1j) + (dc2k - dc1k) \qquad (5)$$

The last term of this equation is the difference in delay of the two clock lines that reach module k (alpha.kk), while the next to the last term is the difference in the arrival of the phase-one clock at modules k and j. This last term is commonly referred to as clock skew (delta.kj). Note that alpha.kk is a result of using a two phase clocking scheme and might not be present with a single phase design. Thus

$$alpha.kk = dc2k - dc1k \qquad (6a)$$

$$delta.kj = dc1k - dc1j \qquad (6b)$$

T2 and T3 can now be rewritten as:

$$T2 > \phi 1 + \phi 12 + dXij + delta.ij + alpha.ii \qquad (7)$$

$$T3 > \phi 1 + \phi 12 + dAj + delta.Aj + alpha.jj \qquad (8)$$

where

$$alpha.ii = dc2i - dc1i \qquad (9a)$$

$$alpha.jj = dc2j - dc1j \qquad (9b)$$

$$delta.ij = dc1i - dc1j \qquad (9c)$$

$$delta.Aj = dc1A - dc1j \qquad (9d)$$

Now $(\phi 1+\phi 12)$ represents the amount of time necessary for the capture of the X, Y and A values plus the time to compute the new value of Y plus the interphase time. Let us represent this as a single delay, dcompute. Furthermore, the terms dA, dX, and dY represent intermodule path propagation delays, and we will identify them as dpathA, dpathX and dpathY, respectively. Then the three intermodule constraints on the clock period can be rewritten as

T1 > dcompute+dpathY+delta.kj+alpha.kk (10)
T2 > dcompute+dpathX+delta.ij+alpha.ii (11)
T3 > dcompute+dpathA+delta.Aj+alpha.jj (12)

2.1.2 Clock Path Constraints

The clock path consists of a conductor that connects the central clock to each module. When the clock is asserted and deasserted this entire path must be charged to pass above the module threshold voltage and discharged to pass below the module threshold voltage. Since the path is passive, the charge and discharge times can be assumed to be equal and will be denoted as tau.ϕ1 and tau.ϕ2 for the two phases respectively. Then to guarantee that module threshold requirements are satisfied the period of the two clocks must be

T4 > 2tau.ϕ1 (13)
T5 > 2tau.ϕ2 (14)

As a consequence the constraint on the clock period can be specified as

T > max{T1,T2,T3,T4,T5} over all modules. (15)

In addition to this constraint we must also guarantee that the two clock phases do not overlap at the input to any module. That is at each module input:

ϕ12 > 0 and ϕ21 > 0 (16)

Figure 5 shows a timing diagram for module j. From this diagram

ϕ21j = ϕ21+dc1j-dc2j = ϕ21-delta.jj > 0 (17a)
ϕ21 > delta.jj (17b)
ϕ12j = ϕ12+dc1j-dc2j = ϕ12-delta.jj > 0 (18a)
ϕ12 > delta.jj (18b)

Similar expressions for the interphase times at other module inputs yield:

ϕ12,ϕ21 > max{delta.ii,delta.jj,delta.kk} (19)

Each of the parameters in the five constraints (10,11,12,13,14) has a range of values which depend on the variations in the fabrication process, the computation time, and in the interconnection pathways. Let us assume that we have a large array, that the delays take on their full range of values and, because the array is large, there exists an adjacent set of three modules i,j, and k and a coefficient array, A, in which all the parameters have their worst case values. Then the identifying subscripts can be removed from the constraint equations and they can be rewritten as

T > max{dcompute+dpath+delta+alpha,2tau} (20a)
ϕ12 > delta and ϕ21 > delta (20b)

These equations show how clock skew, computation time, intermodule path delays, and clock path charging time, affect the clock period and thereby the data rate. Thus the data rate for the asynchronous array is

DRs <
1/max{dcompute+dpath+delta+alpha,2tau} (21a)

Usually the first term in the maximum expression dominates and under this condition

DRs < 1/(dcomput+dpath+delta+alpha) (21b)

This issue will be examined again in Section 5.

2.2 Asynchronous Control

The one dimensional array shown in Figure 1 can be adapted to incorporate the asynchronous control structure. In a self-timed system events can be thought of as having a fixed position in the sequence domain with no event having to occur at a particular or fixed time. Self-timed systems are an interconnection of components where each component performs a step in the desired computation. To maintain the order in the sequence domain necessary to perform the computation, a signal is necessary at the input of a component to initiate the computation step, and a signal is necessary at the output of the component to mark the completion of the computation step.

Considering a particular module j in the one dimensional array, the computation performed by the module is the inner product step (1). Each module must have three data input lines, and two data output lines. The control lines necessary for each data input (output) line to achieve self-timed operation are:

(i) An input (output) line which signals the availability of data on the particular data line, referred to as the data available line, DA.
(ii) An output (input) line which signals the completion of use of data on the data line, referred to as the data acknowledge line, A.

Figure 6 shows the asynchronous array module and associated control signals.

2.2.1 A Petri Net Specification of the One Dimensional Array Module

Because of the concurrency in the control and data flow in the array module, it is not possible to formally specify the behaviour of such a system by a state table. Instead a Petri net is used since this can preserve all the concurrency in the processing steps and should maximize the data rate through the module. The following characteristics of the asynchronous module (Figure 6) are important in order to construct the Petri net:

(1) New data can be made available to the module on the right on XDAR if the acknowledge of the previous data is available on XAR and data is available from the module on the left on XDAL.
(2) The computation process in the module consists of two distinct steps, a multiplication followed by an addition. The multiplication step can begin as soon as data on XDAL and ADAT are available and the previous addition step is complete.
(3) The acknowledge signal AAT for ADAT can be sent after the multiplication step is complete. The acknowledge XAL for XDAL is sent only after the multiplication step is complete and the acknowledge to the data available signal XDAR on XAR has been received.
(4) The addition step can begin as soon as the data required for this step is available. That is, after the multiplication step is complete and data is available on YDAR.
(5) New data can be made available to the module on the left when the addition step is

complete and when the acknowledge to the previous data on YDAL has been received on YAL.

(6) The acknowledge signal on YAR is sent after the addition step is complete and the acknowledge signal YAL has been received.

Ordering YAR after YAL and XAL after XAR becomes necessary to avoid generating a new output before the old output has been used. For example, if we send the YAR signal independent of the YAL signal, new data will become available for the addition step and a new result could be generated before the old data has been used, that is, the addition step could end before the acknowledge signal is received on YAL. Similar reasons necessitate the ordering of XAL after XAR.

The Petri net for the array module j is presented in Figure 7 and can be divided into essentially identical upper and lower halves. In the one dimensional array, the module k is one step ahead in the computation process than module j because module k starts the computation process before module j. The Petri net therefore represents the interaction between two adjacent modules, and between two adjacent computation steps. This results in the two identical halves of the Petri net.

The delay between successive computations in module j can be obtained by analyzing the paths in the module and determining the maximum "loop" delays. To be as general as possible, this analysis should take into account the concurrency within the module. Such an analysis, however, becomes quite involved and is not consistent with the derivations for the synchronous control structure. If we remove the concurrency from within module j, the path having the largest delay is shown with its associated delays by the dashed line in Figure 7. The path involves propagation delays that are internal to the module (subscript i), that are external to the module, that is, intermodule propagation delays (subscript e), and the delay due to the computation.

The intermodule path delays will, in general, be much larger than the internal propagation delays. We will therefore assume that the internal delays have been included in the intermodule path delays. The delay through the module is then given by

$$dasync = dcompute + 2dpath \quad (22)$$

where dpath is the intermodule path delay and dcompute is the delay in the computation. The data rate for the one dimensional array is then

$$DRa = 1/(dcompute + 2dpath) \quad (23)$$

3.0 DATA RATE COMPARISON

The data rates for the synchronous and asynchronous control schemes can now be compared by examining equations (21b) and (23). The asynchronous data rate is greater than the synchronous data rate when :

$$delta > dpath - alpha \quad (24)$$

If the clock lines for both phases are laid out on the chip parallel to each other so that they have equal length paths and drive identical circuits, then delta will be approximately equal to alpha and (24) becomes:

$$delta > dpath/2 \quad (25)$$

The synchronous/asynchronous design decision can now be presented as shown in Figure 8. Once information about specific implementation parameters is available, the designer can calculate delta and dpath. Depending on the region of the design space in which the proposed implementation falls, the designer can then determine whether a synchronous or asynchronous design has a higher data rate. Note that for a given path delay, a larger clock skew will result in asynchronous control being preferable, while a smaller clock skew will result in the synchronous scheme being preferable. The reason for this is that a larger clock skew requires longer clock periods for reliable operation, and this in turn slows down the operation of a synchronous system.

On the other hand, for a given clock skew, longer path delays result in synchronous control being faster. This is due to the fact that with asynchronous control, signals must travel twice along dpath in a request/acknowledge fashion, while in the synchronous case only a single dpath delay is necessary. Thus, long path delays penalizes the asynchronous control architecture more than synchronous one.

4.0 PATH DELAYS AND CLOCK SKEW

Let the linear array be implemented using NMOS technology with a collection of N' modules. Assume that the arrays of interest are sufficiently large so that several chips, each having N modules, will be required to implement the entire N' network. Figure 9 illustrates this for the case of N=2. The chips themselves are assumed to be placed on a printed circuit board and interconnected via printed circuit wiring. In this section, the factors that affect the path delays and clock skew are examined and simple expressions for each of them are developed.

4.1 Path Delay

There are three intermodule path delays of concern, dX, dY and dA and these paths may be between adjacent modules on the same integrated chip, between adjacent modules on different chips, or (in the case of dA) between the source of the A coefficients and the modules.

Assume for the moment that modules within a chip can be positioned so that adjacent modules are in close physical proximity. In such a situation, path delays between these modules will be very small compared to other delays and can be ignored. These other delays (i.e. dA and interchip dX and dY delays) are determined by the resistance and capacitance of the path involved. For the case of dX and dY paths between chips, a chip/board layout can be designed which results in chips containing adjacent modules also being in close physical proximity on the board. Since the interconnection path in this case will be short and will use metal conductors, its resistance is

small and can be neglected. Thus the propagation delay in this case is determined by the ratio of the capacitance of an elemental gate, Cg, and the capacitance of the interconnection line, Cl. If one uses an exponential buffer this delay is given by the expression

dpath = d*e*ln(Cl/Cg) (26)

where d is the delay associated with an elemental transistor, and e is the natural logarithm base.

Determining the path delay associated with dA is more complex since it involves the A matrix elements. For simplicity we assume that these elements are loaded into local memories before processing begins. If, for example, A represented the coefficients of a fixed digital signal processing filter, read only memories might be used. These memory chips are placed so that they are physically adjacent and directly above each systolic chip as shown in Figure 9. The path delay, dA, in this case is similar to the dX and dY delays between chips and is also given by (26).

Once the coefficient memories are loaded, the processing can begin. If a new A matrix is required, it is possible to overlap this loading operation with ongoing processing, but the details of this are not pursued here. Note that another interesting design option is to place these local memories within the chip itself.

4.2 Clock Skew

The clock skew and alpha parameter have been defined earlier. With the appropriate clock line layout the clock skew and alpha will be about the same, hence we only discuss the clock skew. The skew of concern here is generated within the chip. That is, the assumption is made that the clock is presented with zero skew to each chip on the board.

Consider the simple model in which two modules M1 and M2 are on the same chip, are adjacent, and are driven from a common point P, with clock lines within the chip from P of lengths L1 and L2. If the clock is asserted at t = 0, the clock skew is the difference in time between when M1 and M2 respond to the clock assertion. This difference in response is determined by four factors:

1. Differences in the line lengths L1 and L2.
2. Differences in delays through any active elements inserted in the lines (e.g. clock buffers).
3. Differences in the threshold voltages of the two modules M1 and M2.
4. Differences in the line parameters (e.g. resistivity, dielectric constant).

Factor 1 can be made negligible by making the distances from the clock signal entry point on the chip to each module equal. One way of achieving this is by laying out the clock signal paths using a binary tree distribution. An example of a layout for a four module chip is shown in Figure 10.

If one assumes that the synchronous module only requires unmodified phase-one and phase-two clock signals, then, given sufficient off-chip drive capabilites, there is no necessity to buffer the clock signals on the chip. When module implementations are non-pipelined, requiring only phase-one and phase-two clock signals seems reasonable. Factor 2 is thus eliminated.

A model for the clock skew based on the last two factors has been developed and reported on previously for a full binary tree [WANN83]. The skew can be found in terms of the maximum and minimum time constants of the entire clock tree (MAX[RC] and MIN[RC]), the maximum and minimum values of the threshold voltage of a typical logic gate (MAX[Vt] and MIN[Vt]), and the supply voltage Vdd. The results of that derivation gives the clock skew as

delta = MIN[RC]*ln(1 - MIN[Vt]/Vdd) - MAX[RC]*ln(1 - MAX[Vt]/Vdd) (27)

4.3 Clock Distribution and Time Constant

Assume that the binary tree clock distribution scheme shown in Figure 10b is used, the minimum feature size is Lam (in microns), and that modules are square. The time constant for a full binary tree has been derived in [KUNG82C] and verified by SPICE simulation to be:

RC = 1.43(N**3)(3 - 2/N)R0C0

where R0 and C0 are the resistance and capacitance of a leaf node level clock line and N is the number of modules on the chip (N is a perfect square equal to 2**n, where n is an integer).

Note that distribution of the two-phase clock requires two such binary trees - the second tree can be constructed by merely displacing the first tree in the vertical and horizontal directions by the minimum line separation. The clock should be distributed using metal wherever possible. Some short sections of diffusion will be necessary, however, in order to bridge intermodule communication, power, ground and reset lines (assuming only a single layer of metal is available). Let 10% of the clock line be in diffusion. Then R0 and C0 can be expressed in terms of Rd, the resistance of a diffusion line expressed in ohms per square, Cm and Cd, the metal and diffusion line capacitances expressed in terms of picofarads per sq. micron and Wm and Wd, the widths in microns of the clock line implemented in metal and diffusion respectively.

R0 = (Rd/sq)(Ld/Wd) (29a)

C0 = (Cm/ar)WmLm + (Cd/ar)WdLd (29b)

Mead and Conway [MEAD80] show that the minimum width of a diffusion conductor should be 2Lam and that of a metal conductor should be 3Lam. Typical values for the constants representative of current fabrication technology are:

$Cm = 0.3 * 10^{-4}$ pf/um^2, $Cd = 10^{-4}$ pf/um^2

Rd = 20 Ohms/sq

The time constant (R*C) of the clock line is then given by:

$$RC = 1.158*10^{-4}(3 - 2/N)E^2 M^3$$

where E is the length of the leaf node, it is assumed that the clock line is distributed through the entire length of the module, and M is the square root of N. Note the above expression does not take into account the time constant for charging and discharging that part of the tree external to the chip. Our calculations indicate that when the entire network is on a single board this time is less than 10% of the on-chip charge/discharge time and thus is not included in this development.

5.0 EXAMPLE

Consider a processor constructed on a single printed circuit board containing packaged chips with copper printed circuit connections between packages. The pin capacitance for this type of construction is about 4 pf, the capacitance, Cg, of an elemental NMOS gate is about 0.02 pf, and the delay, d, associated with an elemental transistor is about 2 nsec. Since the metal data lines between chips are short, the principal contribution to the load capacitance, Cl, will be the pin capacitance. The path delay, dpath, can then be calculated from (26) as dpath = 28.8 nsec.

In order to determine the clock skew from (27), RC must be evaluated using (30). To determine E and M the area requirements for the module, and the number of such modules which can placed on a chip must be estimated. Currently produced chips [MCDO82] require an area of about 2 square millimeters for implementation of a parallel 16 bit multiplier (Lam about 2 microns). If we assume that another 2 square millimeters is needed for the adder, registers, and associated control, then about sixteen modules would fit on a 1 cm square chip. Taking E as 1/8 cm, RC can be found from (30) to be 29 nsec. and thus 2tau is 58 nsec. The clock skew can be found from (27). Take Vdd = 5 volts, the threshold voltage = 2.5 volts, and a variation of +/- 20% due to fabrication. The resulting clock skew is 20 nsec.

In this example the clock skew (20 ns) is greater than dpath/2 (14.4), and an asynchronous architecture would be faster than a synchronous one. To determine how much faster, the ratio DRs/DRa can be evaluated from (21b) and (23). This can be done once the time for the computation (multiplication and addition) is known. Note that the larger the value for dcompute, the less the difference between the two strategies. For instance if dcompute =150ns in the above example, the difference between the two data rates is less than 10%. Note also that 2tau (58 ns) is a good deal less than dcompute and the other terms in (21a). That is, the time to charge the clock tree is not a limiting factor in this case.

6.0 SUMMARY AND CONCLUSIONS

This paper has presented models for comparison of synchronous and asynchronous control structures in a one dimensional regular array environment. A detailed description of the control architectures was given. The timing sequences required for a two phase clocked system were provided, and a Petri net description of the asynchronous control specified. From these models the data rate expressions associated with each control scheme were derived.

Both expressions indicate that the data rate is inversely proportional to the computation time. The asynchronous structure data rate is also related to two times the path delay while the synchronous data rate involves only a single occurrence of the path delay. Thus large path delays involving the movement of X, Y or A data tend to reduce performance of the asynchronous approach more than the synchronous approach. On the other hand, the synchronous control is dependent on the clock skew and as the clock skew increases, the performance of the synchronous scheme falls. Given the importance of path delay and clock skew, a module and clock line layout was proposed which equalizes clock line lengths. Finally, the time to charge the clock tree also limits the performance of the synchronous design. For situations where large trees and short compute times are present, this may be the dominant factor limiting performance. In a given situation, the performance of the two control schemes is dependent on the relative magnitudes of these parameters.

An example was presented where these parameters were evaluated with the modules being implemented in NMOS with a 2 micron feature size. Assuming a computation time of 150 nsec, it was found that while the asynchronous system had the higher data rate, it was within 10 % of the synchronous rate. The reason for the close performance of both schemes is that computation time plays a major role in determining the overall rates, and that as long as this time is significantly larger than skews, delays and tree charging times, the relative performance of the two schemes will be comparable.

7.0 REFERENCES

CAPP81 Cappello, P.R. and Steiglitz, K. "Digital Signal Processing Applications of Systolic Arrays", in VLSI Systems and Computations, Ed. by Kung and Steele, Computer Science Press, pp 245-254, 1981

CLAR67 Clark, W.A. "Macromodular Computer Systems", AFIPS Proc., SJCC April, 1967

DIGI81 Digital Equipment Corp. MICROCOMPUTERS AND MEMORIES, DEC, Maynard, Mass, 1981

KUNG80 Kung, H.T. and Leiserson, C.E. "Algorithms for VLSI Processor Arrays" Section 8.3 of Introduction to VLSI Systems, C. Mead and L. Conway, Addison-Wesley, 1980

KUNG82A Kung, S.Y. et.al. "Wavefront Array Processor: Language, Architecture, and Applications", IEEE Trans. on Comp., Vol.C-31, No.11, pp 1054- 1066, 1982

KUNG82B Kung, S.Y. (editor) PROC. OF USC WORKSHOP ON VLSI AND MODERN SIGNAL PROCESSING, USC, 1982

KUNG82C Kung,S.Y., R.J. Gal-Ezer "Synchronous vs. Asynchronous Computation in VLSI Array Processor," Proc. SPIE, Vol. 341, May 1982

MCDO82 McDonough, K., et.al. "Microcomputer with 32 Bit Arithmetic" Electronics, Feb.24, 1982

MEAD80 Mead,C. and Conway, L. INTRODUCTION TO VLSI SYSTEMS, Addison- Wesley, Reading, Mass, 1980

SEIT79 Seitz,C.L."Self-Timed VLSI Systems",Proc Caltech. Conf. on VLSI Jan. 1979

SNYD82 Snyder,L. "Introduction to the Configurable, Highly Parallel Computer", COMPUTER, Vol.15, No.1, pp 47-56, 1982

SUTH79 Sutherland, I.E., et.al.,"The TRIMOSBUS", Proc. Caltech Conf. on VLSI, Jan 1979

WANN83 Wann,D.F. and Franklin, M.A. "Asynchronous and Clocked Control Structures for VLSI Based Interconnection Networks", IEEE Trans. on Comput., March 1983

YEN81 Yen, D. and Kulkarni,A. "The ESL Systolic Processor for Signal and Image Processing", Proc. Comput. Soc. Workshop on Computer Arch. and Image Database Management, Nov. 1981

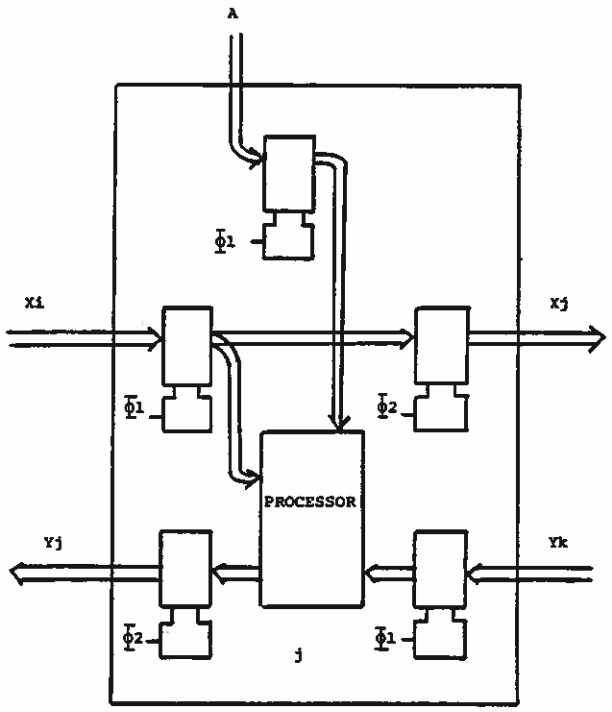

FIGURE 2

REGISTER/PROCESSOR ARRANGEMENT FOR MODULE j

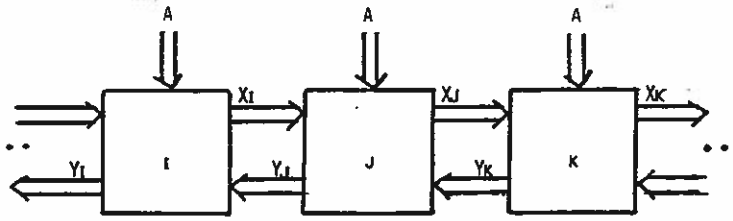

FIGURE 1

ONE DIMENSIONAL SYSTOLIC ARRAY

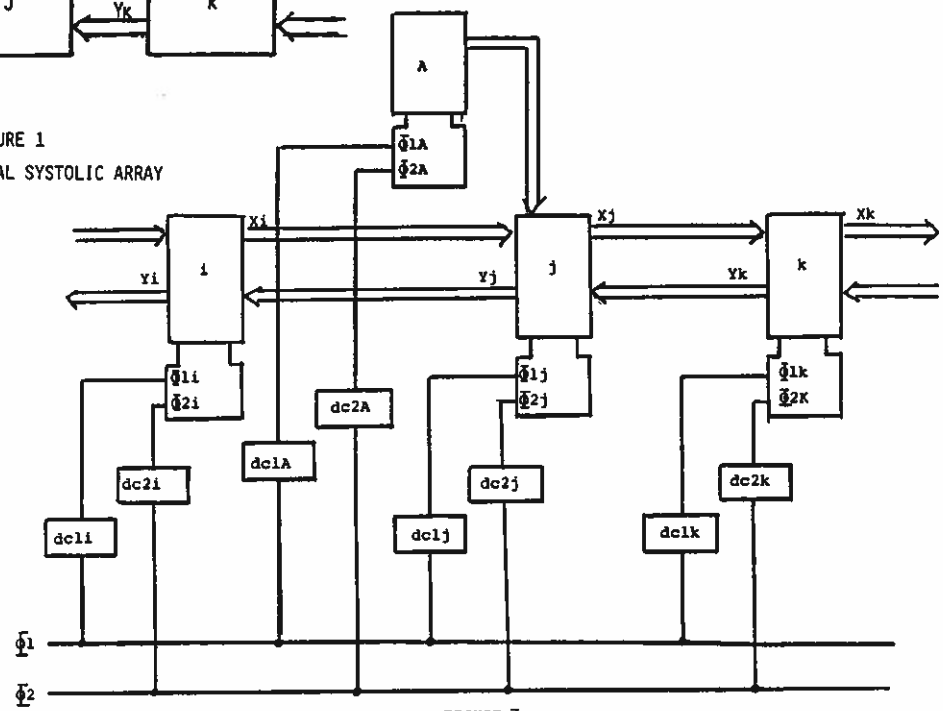

FIGURE 3

CLOCK DISTRIBUTION TO i,j,k, AND A

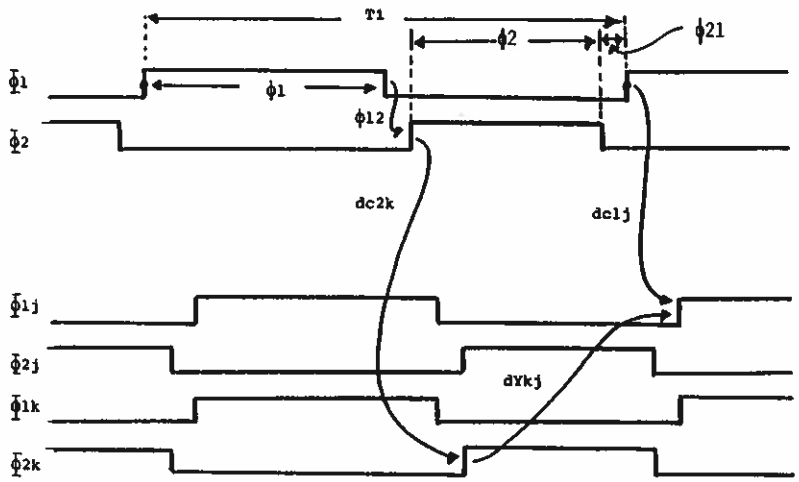

FIGURE 4

TIMING DIAGRAM FOR Yk TO j

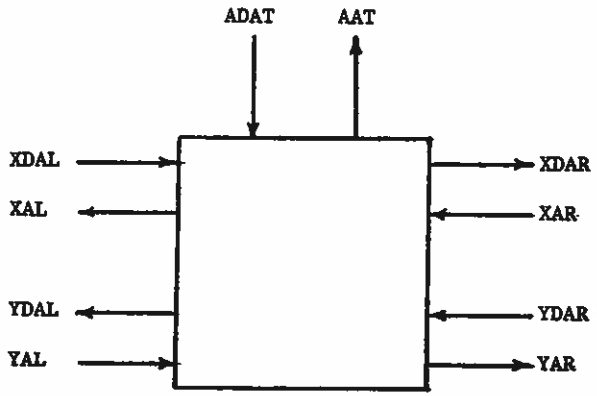

Figure 6: Control signals of the asynchronous systolic array module.

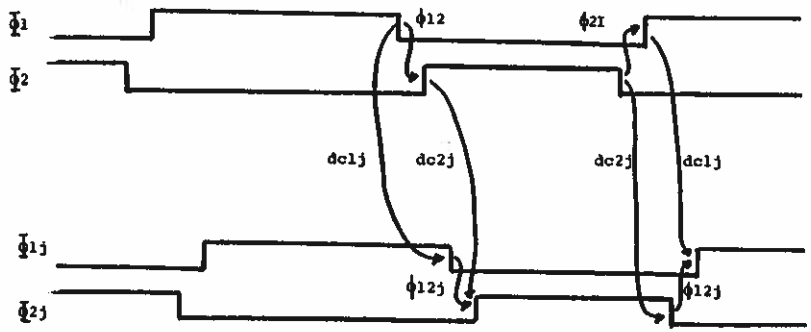

FIGURE 5

TIMING DIAGRAM FOR INTERPHASE CONSTRAINTS

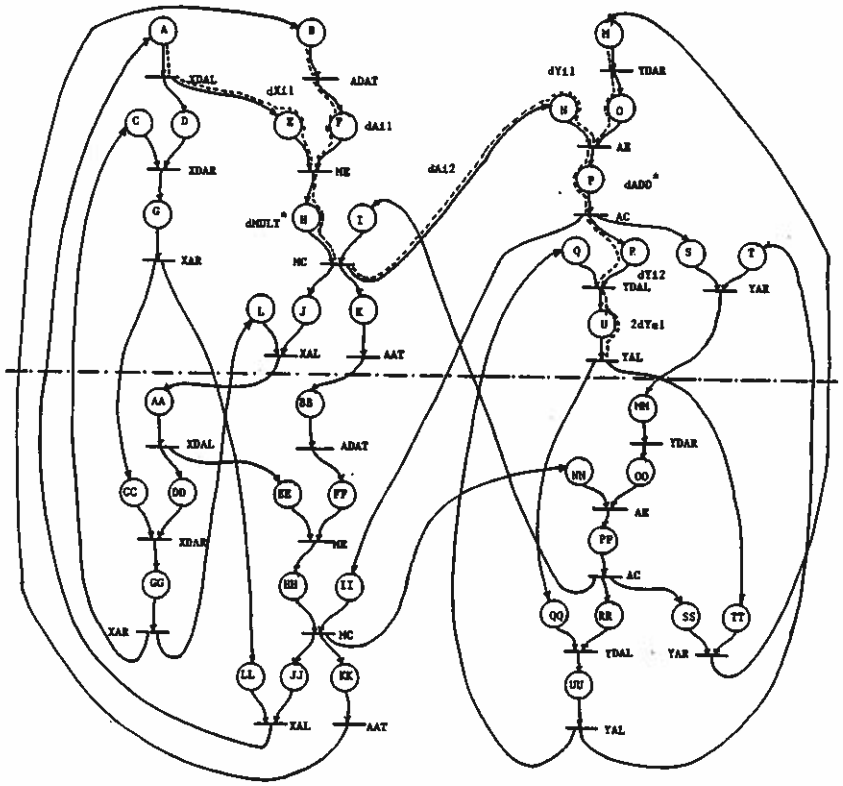

Figure 7: * dcomputer = dMULT + daDD

Petri Net for the One Dimensional Asynchronous Systolic Array Module

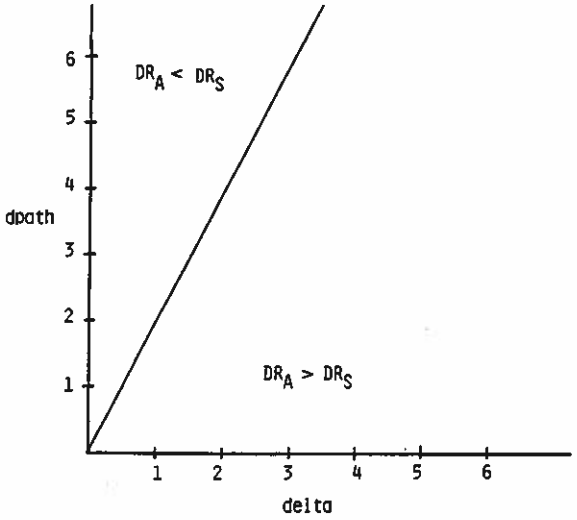

Figure 8: Design space as a function of clock skew and path delay for asynchronous-synchronous architectures.

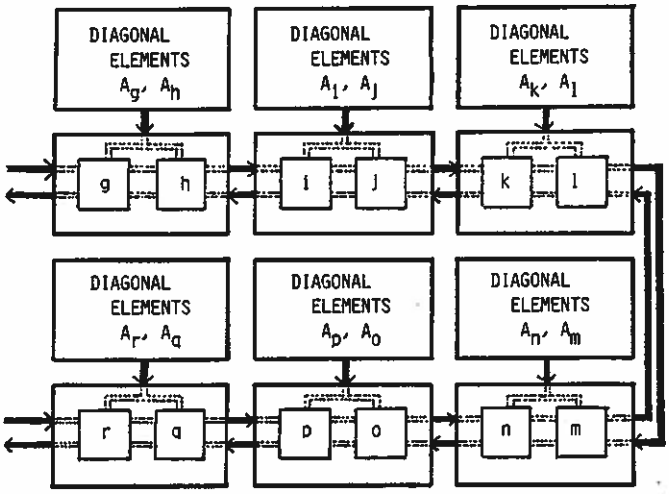

Figure 9: Module-Chip-Board Layout for Small Linear Systolic Array

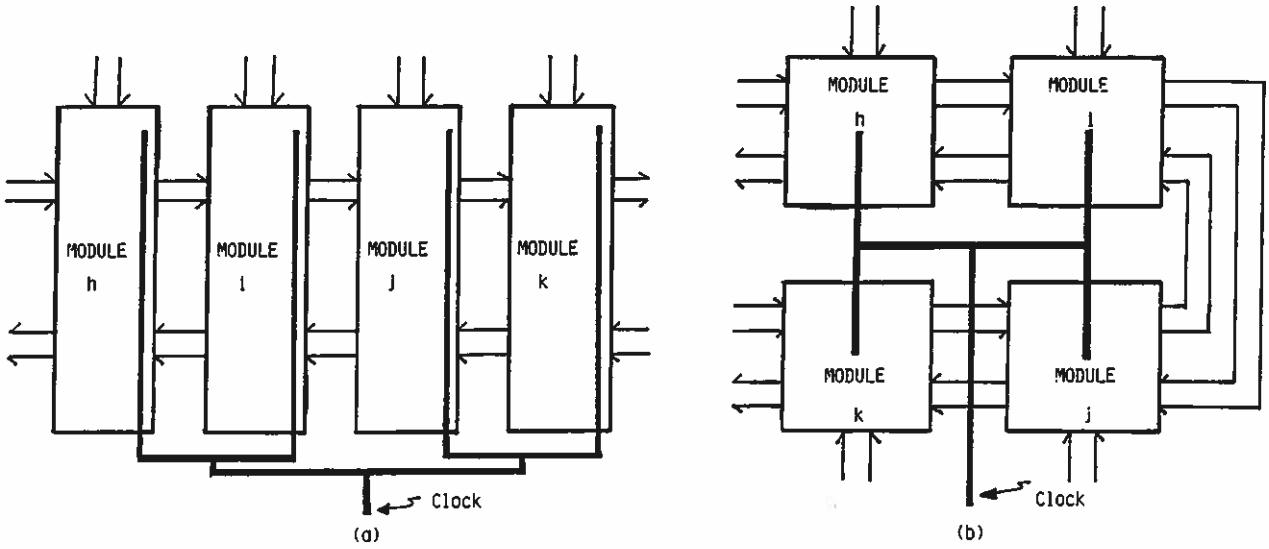

Figure 10: Clock Distribution Within a Chip
a) Rectangular Modules, b) Square Modules