Estimations of Power Consumption in Asynchronous Logic as Derived from Graph Based Circuit Representations

L. Lloyd, A. V. Yakovlev, E. Pastor, A. M. Koelmans
Department of Computing Science,
University of Newcastle upon Tyne, NE1 7RU, England.

May 27, 1998

Abstract

We present a technique for the estimation of power consumption in asynchronous circuits through the modelling of transition switching activity. Unlike most existing techniques for analytic (non-simulation) power estimation that use reachability state traversal and Markov chain analysis, our method is based on an invariant analysis of Petri Net models using matrix representations. This approach is in general more efficient than Markov chain analysis, due to the avoidance of state explosion, but may lose accuracy for some classes of nets.

The asynchronous circuits under analysis are speed-independent designs that are synthesized from Signal Transition Graph descriptions using the tool Petrify.

Keywords: Asynchronous Logic, Invariant Analysis, Petri Nets, Power Estimation, Signal Transition Graphs.

1 Introduction

The past decade has seen an exponential growth in the technological development of computing systems. We now have chips that contain multi-million transistors that can operate at frequencies in excess of 200MHz and yet which are common workplace machines. In general it is the case that as logic features are implemented in silicon at increasingly smaller dimensions there is a corresponding escalation in the frequency at which that logic may operate. These increases in frequency are enabled by a proportional increase in power usage and it is at this point where we begin to observe problems. In these new faster and denser chip designs we see very high power consumption which has the consequences of causing these devices to have very high operational temperatures.

In any integrated circuit the issue of how to deal with power dissipation can be a problem that can limit the actual design process of the device itself, especially where portability is a prime design factor. Excessive power consumption can cause a device to overheat, causing performance loss, ultimately leading to device failure. Solutions proposed to address heat dissipation often

involve the use of fans or heat sinks which themselves consume power or just add to the general bulkiness of the product.

The solution to these problems is to ensure that you *design for low power*. This is a task that must be done early in the chip design stage thus allowing a designer to construct a product whilst staying within specific power consumption limits. In order for this process to occur there needs to be support from the very CAD tools that are used to create these designs. Many modern CAD tools are capable of performing layout, routing, verification, synthesis and simulation but only a few are capable of performing accurate power consumption analysis i.e. PowerMill [4].

The types of device that we have described above tends to be of a synchronous nature, will employ combinatorial logic and utilise a CMOS technology. There is an alternative to this main design methodology though that has been promoted over the past decade and which promises a low powered design approach. This methodology is asynchronous logic design. There are many reasons why designers are now proposing that asynchronous logic is the next step forward for use in digital applications but for the purpose of this discussion we are only interested in the following main argument;

- **low power:** A typical synchronous circuit will employ a global clock which means that the majority of that circuit will be active at all times. The continual decrease in CMOS process rules has meant that modern CMOS logic has become much denser and faster to the point where modern microprocessors can be expected to dissipate 20 to 30 watts of power. With the current trend towards even greater miniaturisation these problems of power dissipation will continue to increase. The lack of a clock in an asynchronous circuit means that only those parts of the logic that are requested to perform any processing actions will be powered. This is the point at which we get the notion that asynchronous logic design is inherently *low-powered* in aspect.

When referring to CMOS technologies we are specifically concerned with the nature of the logic elements as used in such designs. CMOS components only draw on a current supply during a logical transition and so when employing an asynchronous design methodology within a CMOS framework we achieve an ideal environment for the design of low-powered microlelectronic devices. But, as mentioned previously, any such design is critically dependant upon the CAD tool used and if that CAD tool cannot perform satisfactory power estimation techniques the benefits of asynchronous logic may be lost simply through the inability to check that a design does not perform an excessive amount of switching activity.

Before effecting any type of power estimation it is important to classify exactly what is meant by *switching activity* and how it can be modelled. If the input vector to any circuit is unknown power estimation becomes an extreme problem because you cannot determine what states that circuit will enter. This problem becomes worse when, if the circuit under test forms part of an embedded system whose input-output behaviour is currently unknown, you cannot directly simulate that circuit with a set of pre-defined vectors. The solution to this problem is to test for all possible input cases but when considering large circuits this becomes practically impossible. This type of problem is referred to as being *input pattern-dependent*.

The main technique that is used to overcome this problem of input pattern-dependency is to describe the set of all possible input signals using *probabilities*. Using a probabilistic vector input method provides a means to estimate power consumption based on the collective presence of all the possible input signals. This type of problem is referred to as being *pattern-independent*.

This paper will set out to describe a high level model based method of power estimation for use in asynchronous circuits that have been derived from graph representations. The approach will employ pattern-independent input vectors which will allow the estimation of the *average* power consumed by a digital circuit. Whilst this method of power estimation is coarse grained, being based on the cumulative summations of transition switching, this very generality allows accurate results to be compiled. One primary advantage to this type of modelling is to allow the simulation of circuits to take place that are independent of design methodology, technology mapping and operational functionality etc.

The remainder of this paper will be set out as follows; section two will be concerned with describing some of the more prominent methods of power estimation in CMOS circuits from both a synchronous and an asynchronous point of view. Speed-Independent (SI) circuits will be discussed briefly in section three together with their synthesis considerations using the tool Petrify. Section four will give an introduction into Petri Nets (PN) and Signal Transition Graphs (STG) and will describe how the method of using firing vectors as derived from a matrix description of a PN, representing the logical behaviour of a circuit, can be used to estimate power consumption. Section five will describe environmental conditions that also have to be taken into account, for example delay models, with section six presenting examples of the synthesis and modelling of a number of asynchronous control circuits. Section seven concludes the paper.

2 Previous Work

The estimation of power consumption in any digital circuit, regardless of the physical level at which such analysis takes place, can be located in the domain of modelling alternatives as illustrated in Figure 1 below:

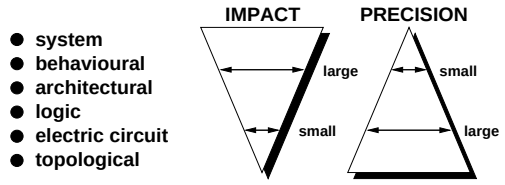

Figure 1: Domains of power estimation.

In general it can be said that the relation between the *impact* of change to a circuit is a monotonically decreasing function of the *precision* to which that circuit can be analysed. In essence this means that, for example, changes to a system level model of a design will have a small effect on the precision at which such a design may be investigated as it is likely that the system model is already of such an abstract nature that such changes will have no appreciable effect. Conversely at the electric circuit level, where it is possible to model very precisely

operational characteristics, small changes to the structure of that circuit, whilst not affecting the precision to which that circuit may be evaluated, may significantly change the quantitative nature of those results.

For the technique of power estimation that is exemplified here we are only concerned with modelling at the logic level of a circuit.

2.1 Power estimation in synchronous circuits

The estimation of power consumption in digital circuits is a subject to which a great deal of investigative research has been applied. These studies have resulted in a number of techniques that can be used to analyse a circuit at various levels of abstraction [14].

Typically the power consumed by a CMOS circuit is said to be comprised of three major components:

$$P = P_{load} + P_{short\ circuit} + P_{leakage}$$

In current CMOS technologies the load power is the factor that dominates the dissipation of energy. The *load current* of a gate is the current that charges and discharges the fan-out capacities of that gate. In general the energy that is transferred during any charging or discharging process is:

$$E_{charge} = \tfrac{1}{2} \cdot V_{DD} \cdot C_{load} \cdot \Delta V$$

where V_{DD} is the supply voltage, C_{load} is the capacity to be charged and ΔV is the voltage swing at that capacity. The total energy consumed for all n transitions of a gate is:

$$E_{total} = \tfrac{1}{2} \cdot C_{load} \cdot \sum_{i=1}^{n} \Delta V_i$$

As the power dissipation of any circuit is intrinsically dependent upon the functional application of that circuit any power analysis technique applied requires a familiarity with the typical activity of the signal inputs. For the evaluation of a circuit at the logic level we have already mentioned how the simulation vectors must be either one of two types; input pattern-dependent or pattern dependent [15].

With a pattern dependent method input vectors can be exhaustive simulation based or application based. Using the application based approach knowledge about the average switching activity is used to allow tailored input vectors to be applied as simulation inputs. The exhaustive input vector method requires the simulation of all possible transitions of a circuit. Consequently this means that for an m state device with n inputs there is the requirement of:

$$m \cdot 2^{(2 \cdot n)}$$

input vectors that must be generated and subsequently simulated. For many systems this number becomes too large to be practically useful [6].

When employing a pattern-independent analysis technique for power estimation either a symbolic simulation or a probabilistic simulation approach can be used. In symbolic simulation the internal activity of nodes in a netlist is calculated as a function based on statistical properties of the primary inputs [5]. Essentially this means an equation is produced for each internal input and output node that describes a relationship between the switching activity of that node and the primary inputs.

In probabilistic simulation the statistical properties, as collected from analysis of the primary inputs [19], are propagated through a netlist in order to obtain the switching activity of all the nodes in that netlist.

2.2 Power estimation in asynchronous circuits

There are many theories and established methodologies for the design and implementation of asynchronous circuits and systems. Some notable examples of the techniques that have been developed with the aim of power estimation in asynchronous logic circuits are briefly described in the following.

In [12] power estimation was carried out by using PN based abstractions of self-timed circuits where the analysis technique was to employ *discrete time Markov chains* embedded within reachability graphs of PN's. The average energy consumed was determined by statistically estimating the duration of operations of a circuit. This work was the attempt to integrate both synchronous and asynchronous power estimation techniques and it is worth noting that the results gained by this method were accurate to within 10% of the figures as produced by a contemporary SPICE analysis.

An energy level model of a high level circuit specification, based on a *Communicating Hardware Process* description, was implemented in [18]. This model gave approximate energy expenditure results but the fact that the technique could be employed very early in the circuit design process meant that trade-offs could be made between energy usage, area considerations and delays. This specifically allowed an architecture to be developed that was by nature intrinsically low-powered.

A measure for the estimation of the energy consumed by burst-mode control circuits was proposed in [3]. This work presented two strategies that were based on a stochastic analysis of a number of simulations that could be used to quantify the energy expended by external signal transitions. The first scheme was used to derive mathematical bounds of energy consumption that compensated for the effects of hazards for all the functional and environmental conditions under which a circuit could operate. The second scheme employed these bounds within a fixed-delay simulation context for the accurate estimation of power within burst-mode control circuits. Results gained from employing this technique, e.g. on the simulation of a 400 gate second-level cache-controller, revealed that less that 5% of the energy consumed per signal transition was attributable to hazards.

In [2] a method for the dissipation of energy in SI circuits was described. This approach performed traversals on sequential signal transition graphs, the topology of which allowed a

Markov chain analysis to be carried out. This Markov chain analysis gave rise to a system of linear equations that were used to determine the average energy per external signal transition and thus could provide power consumption statistics for both high-level circuit operations and for circuits incorporating delay specifications.

A basic description of the work to be presented here is concerned with the use of algebraic techniques to perform analysis of graph representations of asynchronous control circuits. These asynchronous circuits are speed-independent designs that can be interpreted as either high-level logic specifications or as synthesized Petrify netlists [8]. Essentially the matrix representations of PN's that correspond to these netlists are analysed using linear algebraic techniques. In context with other work we have similar methods of logic representations as Beerel et.al. in that graphs are used to describe circuit specifications. The work described here shows advantages over the methods of Beerel in that linear algebraic techniques are employed as opposed to Markov chain analysis as the later is often concerned with reachability analysis which at times may result in excessive complexity of problem resolution.

With regards to the work by Tierno et.al. which presents an algebraic analysis of a high level specification language we can be seen to be expounding a contemporary technique of power estimation by using a transition counting approach. In summation this work can therefore be seen to be an amalgamation of a number of current techniques that allows the estimation of power consumption in asynchronous circuits.

3 Synthesis of Speed-Independent circuits

The assumptions that are made in any SI circuit are that all gate delays are unbounded whilst all wire delays will be negligible with respect to those gate delays. Every input vector to an SI circuit will be acknowledged thus allowing the surrounding environment to apply a subsequent vector change. For all the possible permutations that an input vector may describe that relate to a particular state an SI circuit will generate a single or a number of output signals. This is an important property of SI circuits in that this guarantees *hazard-free* operation.

Several methods for the synthesis of SI circuits have been presented. These approaches have been based around mechanisms of how to implement output signals that are produced from circuits constructed from particular types of gate libraries. In [1] networks of gates were used that drove Muller-C elements in order to produce each specific output signal. The use of complex gates to drive output signals was promoted in [7] with later work, based on the decomposition of these complex gates, being described in [11]. More recently the tool petrify [8] has been designed to take a graph based representation of a circuit as input and produce a synthesized SI circuit as a result. Circuits synthesized using petrify can be either complex or decomposed logic gate implementations, based on various technology mapping options, that retain the originally specified input-output behaviour.

Petrify will solve the correctness criterion applied to an STG specification that is necessary for the implementation of SI circuits of: *boundedness*, which assures that the circuit can be im-

plemented with a finite number of logic components, *semi-modularity*, which ensures consistency of output signals giving rise to a hazard-free circuit and *complete state coding*, which confirms that the net description of the circuit is actually implementable as a SI circuit.

In addition to the synthesized circuit that petrify generates an STG of that circuit is also produced that may include extra state signals and which is guaranteed to be live, safe and place-irredundant.

4 Power estimation using graph representations of circuits

4.1 Introduction

Traditional methods for the estimation of power in asynchronous circuits have involved some form of *traversal of states* that have been extracted from a circuit specification created using some synthesis tool. Prime examples of this can be found in the work by Kudva and Beerel, [12] [2], in which Markov chain analysis techniques underpin the power consumption methods employed.

The method of power estimation to be described here proposes improvements to these existing techniques by allowing such an analysis to be performed *before* the synthesis of a circuit is carried out. By taking the behavioural STG specification and creating matrix representations of that circuit we can perform a structural analysis based upon invariants that are extracted from those matrices. This approach removes the need to perform any kind of state traversals for power estimation, specifically a Markov chain analysis which may have incorporated *state explosion* problems. A designer can readily reject a number of initial designs of a circuit before any synthesis procedure is carried out, see Figure 2.

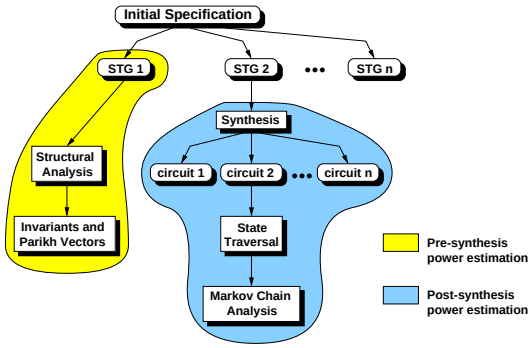

Figure 2: Comparison of Power estimation techniques.

Once a set of invariants have been derived from the matrix analysis we perform the task of annotation of power to the transitions in those invariants. Simulation of the circuit is then the probabilistic execution of a number of *firings* of that circuit with the resultant power consumption being a summation of the power values as associated with the *fired* transitions. This method allows us to perform an average measure of the power consumed by the execution of particular cycles of activity.

One important criteria to note here is that the models under question all have the notion of

incorporating a *home state*. Essentially this home state is a marking of a graph such that after some firing, corresponding to the execution to competition of some task in the circuit, we end up again at the home state. There is no notion of multiple initial states that could give rise to cyclic liveness, or non-terminating activity, within those nets.

The method of power estimation to be detailed here has been described as being the summation of power as consumed by the logical transitions of a circuit. In order to *count* these transitional changes a means is required that can represent all the states that that circuit can enter into. Consider the example of an abstract circuit as illustrated in Figure 3 below:

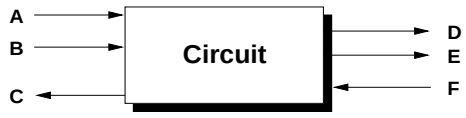

Figure 3: Abstract circuit interface configuration.

A PN that may describe the behaviour of this logic circuit can be seen in Figure 4:

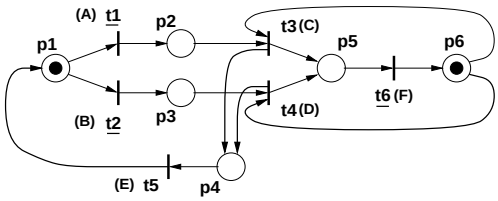

Figure 4: Behavioural specification of the logic circuit as a PN.

In order to perform any analysis on this net we need to create what is known as the *matrix view of a Petri Net*.

4.2 Matrices and firing sequences

STG's and PN's have received much attention over the past few years as being ideal design methodologies for the specification and analysis of asynchronous logic circuits [7, 20, 8]. A PN is a graphical tool that can be used for the analysis of both dynamic and concurrent systems [13]. Any PN is a tuple (representing a graph) such that $\mathbf{P} = (S, T, F, \mu_0)$ where S = the set of vertices that represents the state components (places) of the graph, T = the set of transitions (or actions) that can be performed, F = the flow relation, defined as $F \subseteq (S \times T) \cup (T \times S)$ (both S and T are finite disjoint sets) and μ_0 = the initial state marking of the graph, defined as a function $\mu_0 : S \rightarrow \aleph$. In brief, tokens flow around a net representing events which in circuit terms represents signal changes.

One possible method for the analysis of a PN is based upon the *matrix view* of that net [17]. Using this method a PN is defined by two matrices, D^- and D^+, that represent all the output and input functions that are associated with each place in that net. Each matrix will be m rows (for the transitions) by n columns (for the places) and are defined as:

$$D^-[i,j] = (s_j, t_i) \qquad \text{where: s } \odot\!\!\rightarrow\!\!| \text{ t} \qquad \text{(the pre-set of all transitions)}$$

$D^+[i,j] = (s_j, t_i)$ where: t $\mid\!\!\longrightarrow\!\!\odot$ s (the post-set of all transitions)

If we consider that the union of the elements, $S \bigcup T$, are *nodes* of $\mathbf{P}$ then given a node x of $\mathbf{P}$ we can say the set ${}^\bullet x = \{y \mid (y,x) \in F\}$ is the pre-set of x and the set $x^\bullet = \{y \mid (x,y) \in F\}$ is the post-set of x. The elements in the pre-set (the inputs to a place) can thus be seen to be defined by the D^+ matrix whilst the elements of the post-set (the outputs from a place) are defined by the D^- matrix. These definitions allow a PN to be defined as vector forms.

If we now let $e[j]$ be the unit m-vector, which is set to zero except for the jth component, then any transition t_j will be represented by the m-vector $e[j]$. If that transition t_j is enabled in a marking μ, i.e.:

$$\mu \geq e[j] \cdot D^-$$

then the result of the firing of transition t_j in the marking μ is:

$$\begin{aligned} \delta(\mu, t_j) &= \mu - (e[j] \cdot D^-) + (e[j] \cdot D^+) \\ &= \mu + e[j] \cdot (-D^- + D^+) \\ &= \mu + e[j] \cdot D^I \end{aligned}$$

where D^I, the composite change matrix ($D^I = D^+ - D^-$), is more commonly referred to as the *incidence matrix* of a net and can be formally defined as:

if $\mathbf{P} = (S, T, F, \mu_0)$ and $D^I : (S \times T) \rightarrow \{-1, 0, 1\}$ of $\mathbf{P}$ then

$$D^I(t,s) = \begin{cases} 0 & if \quad (t,s) \notin F \quad and \quad (s,t) \notin F, \quad or \quad (t,s) \in F \quad and \quad (s,t) \in F \\ -1 & if \quad (t,s) \notin F \quad and \quad (s,t) \in F \\ 1 & if \quad (t,s) \in F \quad and \quad (s,t) \notin F \end{cases}$$

The usefulness of the incidence matrix is in the determination of reachability, for example when given an initial marking together with a finite sequence of firing transitions, σ, we can calculate which other markings in that net can be reached. Consider the sequence of firing transitions of σ to be:

$$\sigma = t_{j_1}, t_{j_2}, \ldots, t_{j_k}$$

then

$$\begin{aligned} \delta(\mu, \sigma) &= \delta(\mu, t_{j_1}, t_{j_2}, \ldots, t_{j_k}) \\ &= \mu + (e[j_1] \cdot D^I) + (e[j_2] \cdot D^I) + \cdots + (e[j_k] \cdot D^I) \\ &= \mu + (e[j_1] + e[j_2] + \cdots + e[j_k]) \cdot D^I \\ &= \mu + \vec{\sigma} \cdot D^I \end{aligned}$$

where the vector $\vec{\sigma} = e[j_1] + e[j_2] + \cdots + e[j_k]$ is the *firing vector* of the sequence, σ. The ith element of $\vec{\sigma}$, $\vec{\sigma_i}$ is the number of times that the transition t_i will fire in that sequence. This firing vector is therefore a vector of non-negative numbers with $i \in \aleph$ and is more commonly

known as the *Parikh vector of transition firings*, $\vec{\sigma}$ [16]. An example of how these Parikh vectors are used can be demonstrated by considering the PN in figure 4 above which has a D^I matrix as follows:

$$D^I = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 & 1 & -1 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Consider that we have a specific marking of the PN and are interested in finding which transitions will fire in order to reach that marking as determined from some initial starting point. If we are in the initial marked state of **p1, p6** and wish to traverse to a final marked state of **p1, p5** (see Figure 4 above) then we can calculate a difference marking, γ, from:

$$final\ state,\ \mu^{'} = initial\ state,\ \mu + (\vec{\sigma}\ \cdot D^I)$$

giving

$$\gamma = (\mu^{'} - \mu) = (\vec{\sigma}\ \cdot D^I)$$

and so

$$(1,0,0,0,1,0) = (1,0,0,0,0,1) + (\vec{\sigma}\ \cdot D^I)$$
$$(0,0,0,0,1,-1) = (\vec{\sigma}\ \cdot D^I)$$

gives a firing solution $\vec{\sigma} = (1,0,1,0,1,0)$ which corresponds to the sequence, $\sigma = t_1, t_3, t_5$.

4.3 Rank analysis and firing vector solutions

The technique for employing Parikh vectors of transition sequences would appear to be an ideal method for the quantifying of the transitional activity within a digital circuit. An observation can be made though regarding PN analysis when employing this method is that the D^I matrix does not completely represent the structural topology of that net. If we have the case that:

$$\gamma = (\mu^{'} - \mu) = \mathbf{0} \qquad \text{then}$$

$$(\vec{\sigma}\ \cdot D^I) = \mathbf{0} \qquad \text{and so} \qquad \vec{\sigma} = \mathbf{0}$$

Another point to consider is the lack of sequencing information in any Parikh vector $\vec{\sigma}$. If we have the condition that $\vec{\sigma} \neq 0$ then any reachable marking from σ will depend only upon the number of occurrences of each transition and not the sequence in which they occur. There is

no information regarding the causal relationship between the transition occurrences and every permutation of σ, provided that σ is a valid sequence, will give rise to the same marking.

We will show in the following that, based on the *rank* of the augmented transpose matrix $[D^{I^T} \cdot \gamma^T]$, this problem of indeterminate sequencing of ordering is irrelevant when it comes to the summation of power as consumed in any firing sequence, σ. We can define the rank of any matrix, where such a matrix is denoted by A as follows: the rank A is the cardinality of the number of columns that comprise a linearly independent set. As it is the case that rank A = rank A^T then the rank A^T can be equivalently expressed as the cardinality of the set of the linearly independent rows [10].

The Rank analysis of a PN provides the number of *free parameters* that can be used to solve for the number of *independent paths* that exist within that net. As these PN's are live and so by definition give rise to cyclic graphs we have the situation where we have to decide how many paths do we have to traverse or more strictly how many solutions are there for a particular firing sequence between any two nodes in such a net?

By using integer linear programming techniques we can determine the number of solutions in a net by solving the number of linear equations that are present in the augmented transpose matrix. As the number of equations to be solved may be very large it is possible to reduce the complexity of this operation by first performing a standard procedure of Gaussian elimination on the $[D^{I^T} \cdot \gamma^T]$ matrix before any determination of rank is carried out. An algorithm for deriving the number of firing solutions that exist between any two nodes in a graph that is based on an exhaustive rank analysis is listed below:

proc Firing sequences (S, T, F)
  Create incidence matrix, D^I
  Determine difference marking, γ
  Create augmented transpose matrix, $[D^{I^T} : \gamma^T]$
  Guassian eliminate, (GE) $[D^{I^T} : \gamma^T]$ *matrix*
  if $rank\ GE[D^{I^T} : \gamma^T] \neq rank\ GE[D^{I^T}]$ **do**
    no solutions
  else if $(rank\ GE[D^{I^T} : \gamma^T] = rank\ GE[D^{I^T}] = Card(S))$ **do**
    one solution - solve firing vector
  else if $((rank\ GE[D^{I^T} : \gamma^T] = rank\ GE[D^{I^T}]) \leq Card(S))$ **do**
    $no.\ of\ free\ parameters = (Card(S) - rank\ GE[D^{I^T} : \gamma^T])$
    solve for all firing vectors
  end do
end proc

Algorithm 1: Determination of the number of firing vectors in a PN.

By applying the above algorithm to the $[D^{I^T} \cdot \gamma^T]$ matrix as derived from the PN in figure 4 we achieve the following linear equations:

$$t_1 + t_2 - t_5 = 0 \qquad t_2 + t_3 - t_5 = 0$$
$$t_3 + t_4 - t_5 = 0 \qquad t_5 - t_6 = 1$$

As the number of free parameters of this PN, as calculated from $(Card(S) - rank\ GE[D^{I^T} : \gamma^T])$, can be determined to be 2 then by solving these linear equations we obtain the following Parik vectors that are the firing traces between the markings of **p1, p6** and **p1, p5**:

Parik vectors	Firing Transitions					
	t_1	t_2	t_3	t_4	t_5	t_6
$\vec{\sigma_1}$	1	0	1	0	1	0
$\vec{\sigma_2}$	0	1	0	1	1	0

Table 1: Parik vectors firing solutions for the reachability marking $p_1, p_6 \xrightarrow{\sigma} p11, p_5$.

and it can be seen from the above table that the two firing vector solutions as described in section 4.2 for the example PN actually provides the complete set of firing traces for the reachability marking $p_1, p_6 \xrightarrow{\sigma} p_1, p_5$.

4.4 Invariant firings and circuit simulation

A stipulation of the nets to be simulated here is that the marking of such a net will be equivalent to the *initial state* of the corresponding logic. In defining the case for an initial marking we require that any marking, μ, must satisfy the condition:

$$\exists\ \vec{\sigma} \quad enabled\ at\ \mu\ such\ that \quad \mu \xrightarrow{\sigma} \mu' \quad and \quad \mu = \mu'$$

In other words the Parik vector, $\vec{\sigma}$, enabled at μ reproduces μ. This type of Parik vector is referred to as a T-invariant [9] and a simple test to ensure that a vector is a valid T-invariant is:

$$\exists\ \vec{\sigma} \quad such\ that \quad (\vec{\sigma} \cdot D^I) = \mathbf{0}$$

In order to simulate the behaviour of a net we need to apply a sequence of input vectors. These pattern-independent vectors are supplied as probabilities whose task is to exercise the various paths in that net in a manner analogous to the functional operation of the circuit.

At any *choice-point* in a net we can assign a weighting, or probability, that will be used to determine the functional action of the net at that point. In the case of complex nets, where there may be many instances of choice that have varying degrees of courses of action, we require a means of ordering these probabilities so that collectively they will determine specific firing traces. By exploring the underlying graph structure of a PN we can represent the total number of choice points as a finite tree. This tree structure can be *collapsed* to give a range of probabilities that will represent the *minimal* set of firing sequences within that net. For example, if we have a complex net whose probability extracted tree structure can be seen in figure 5(a):

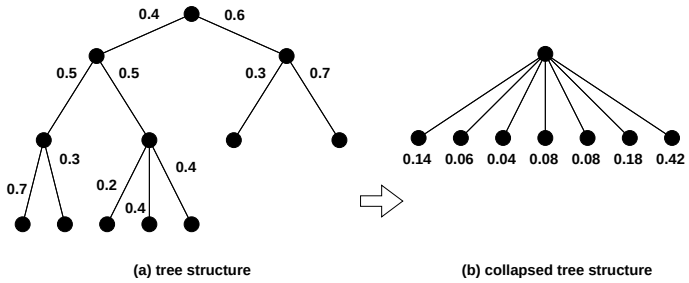

Figure 5: Probability occurrences as tree structures.

then the probabilities of the collapsed net, figure 5(b), represents the minimal set of firings in that net. In order to perform simulation of the net it is now a simple case of assigning the relevant weighting to the Parikh vector that describes that particular firing trace.

This method of determining the set of all probabilistic activities of a net shows significant advantages over the method proposed by Beerel in [2]. In [2] Beerel was concerned with developing a means for the representation of concurrent actions within an STG. Essentially groups of synonymous firings, whose traces led to the same final markings but whose interleaving of actions were dissimilar, were grouped together into a series of equivalence classes. these equivalence classes could then be used to create a *sequential signal transition graph*, SSTG, in which all concurrent behaviour has been serialized. With regards to an invariant analysis of a net we do not have to consider the issue of concurrency because all concurrent activity is *implicit* in the tracing of any firing sequence, $\mu \xrightarrow{\sigma} \mu'$.

The complexity of our method is therefore determined by the size of the PN. Indeed, both the invariant calculation (solving the linear system) and the choice sub-graph traversal are determined by the size of the PN graph. In the state-based techniques, like [12] and [2], the linear system describing the Markov chain is determined by the size of the reachability set, which for a highly concurrent model can be exponentially larger (it should however be noted that work in [2] uses special compression techniques at the state level).

If we have a sequence of probabilistic inputs then the sum of those probabilities must satisfy the condition that:

$$\pi_i \in \prod \qquad \text{then} \qquad \sum_i (\pi_i \in \prod) = 1$$

and as nature of a SI circuit is to execute mutually exclusive events then to sum the energy consumed by one cycle of activity we need to calculate:

$$E_{cycle} = \sum_i (\pi_i \cdot \sum [\vec{\sigma_i} \cdot \Omega])$$

where Ω is the weighting vector, defined as an [n by 1] matrix with n = T, with each element in Ω representing the quantity of power that would be consumed with respect to the firing of a transition in the net.

5 Circuit timing and delay considerations

With respect to SI circuits the delay model best suited for any analysis procedure is the pure delay paradigm. The pure delay model is considered to be the most inaccurate because internal transitions of combinatorial logic will be consumed by the model, having the effect of power estimation values being underestimated because toggle power associated with hazards or glitches are suppressed. As SI circuits do not exhibit significant hazardous behaviour power consumption factors can therefore be considered to be (we consider $P_{leakage}$ to be negligible):

$$P = P_{load} + P_{leakage}$$

In any circuit that employs a mechanism for regulating periodicity of action, for example a clock in a synchronous circuit, we have the facility to couple computational activity with physical time. As the systems under investigation here are time insensitive measures of energy cannot be scaled in the same manner. What can be done though is the comparison of algorithms in order to determine the average energy per cycle of operation:

$$\overline{E}_{average} = \sum_{i} E_{cycle_i}$$

6 Test results

In order to demonstrate the effectiveness of this method of power estimation we shall concentrate on a single worked example. Consider the VMEbus interface controller shown in Figure 6 that has a corresponding STG model, Figure 7:

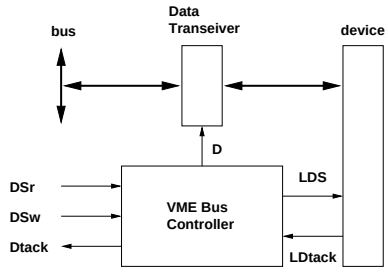

Figure 6: VMEbus interface controller

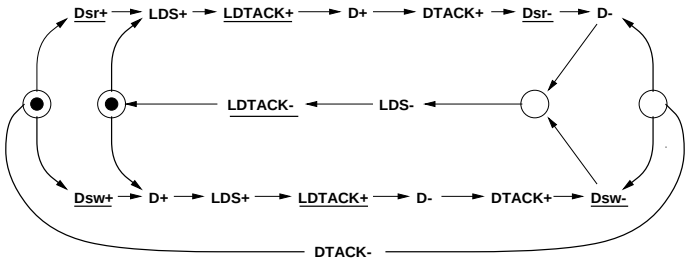

Figure 7: VMEbus interface

Matrix analysis and subsequent Parikh vector extraction provides the following table that describes the complete set of possible firing vectors for this circuit, again using the premise that we always return to the home state after some cycle of activity:

Parik vectors	Firing Transitions																
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}
$\vec{\sigma_1}$	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
$\vec{\sigma_2}$	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1

Table 2: Parik vectors firing solutions.

For the simulation of this circuit we need to determine with what probabilities will these inputs fire. From Table 2 we can see that there are only two modes of activity under which the VMEbus interface controller can operate. These two functions translate to read and write operations and so if we assign probabilities to these functions:

$$\pi_1(\vec{\sigma_1}) = 0.7 \qquad (read\ operation)$$
$$\pi_2(\vec{\sigma_2}) = 0.3 \qquad (write\ operation)$$

and use the weighting vector [1], $\Omega = (14,6,4,4,6,2,14,4,4,2,4,4,6,2,2,6,2)$, then the energy per cycle of operation can be found from:

$$\begin{aligned} E_{average} &= E_{cycle}(\vec{\sigma_1}) + E_{cycle}(\vec{\sigma_2}) \\ &= (0.7 \cdot 36) + (0.3 \cdot 60) \end{aligned}$$

7 Conclusions

We have described a method for the estimation of power in asynchronous circuits based on linear algebraic analysis of matrix representations of net descriptions of those circuits. This method has benefits in that analysis procedures can be carried out prior to circuit synthesis thus avoiding difficulties associated with other traditional modelling techniques such as using Markov Chains.

Through the use of a tool such as Petrify we can rapidly develop a number of STG descriptions based upon the original circuit specification and can analyse these STG's to find those circuits that exhibit desirable properties, in this case low power consumption rates.

This work has been developed in conjunction with the design and development of a number of asynchronous control circuits and in particular the implementation of a 32-bit asynchronous microprocessor. This microprocessor has been implemented using a number of common FGPA products and it is expected that the method of power estimation described here will be used to analyse the complete functional operation of this device.

[1] This weighting vector is derived from a circuit diagram of the VMEbus interface controller as found in [20] and is shown in transpose form.

8 Acknowledgements

Thanks must go to Keith Heron who provided some useful insights into the general switching nature and behaviour of CMOS devices. This work has been partially supported under EPSRC grant numbers 95305908, GR/L28098 and GR/LK70175.

References

[1] P. A. Beerel. **CAD Tools for the Synthesis, Verification and Testability of Robust Asynchronous Circuits**. PhD thesis, Stanford University, 1994.

[2] P. A. Beerel, C-T Hsieh, and S. Wadekar. **Estimation of Energy Consumption in Speed-Independent Control Circuits**. *IEEE Transactions on CAD*, pages 672–680, June 1996.

[3] P. A. Beerel, K. Y. Yun, S. M. Nowick, and P-C. Yeh. **Estimation and Bounding of Energy Consumption in Burst-Mode Control Circuits**. In *Proc. International Conf. Computer Aided Design (ICCAD)*, pages 26–33. IEEE Computer Society Press, 1995.

[4] J. Benkoski, A.-C. Deng, X. Huang, S. Napper, and J. Tuan. **Simulation Algorithms, Power Estimation and Diagnostics in PowerMill**. In *PATMOS'95*, pages 399–410, 1995.

[5] R. Burch, F. N. Najm, P. Yang, and T. N. Trick. **A Monte Carlo Approach for Power Estimation**. *IEEE Transactions on VLSI Systems*, 1(1):63–71, March 1993.

[6] T-L. Chou and K. Roy. **Accurate Power Estimation of CMOS Sequential Circuits**. *IEEE Transactions on VLSI Systems*, 4(3):369–380, September 1996.

[7] T.-A. Chu. **Synthesis of Self-Timed VLSI Circuits from Graph-Theoretic Specifications**. PhD thesis, MIT Laboratory for Computer Science, June 1987.

[8] J. Cortadella, A. Kondratyev, M. Kishinevsky, L. Lavagno., and A. V. Yakovlev. **Petrify: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers**. In *Proc. 11th Conference on Design of Integrated Circuits and Systems (DCIS'96), Barcelona*, pages 205–210, November 1996.

[9] J. Desel and J. Esparza. **Free Choice Petri Nets**. Cambridge Tracts in Theoretical Computer Science 40, Cambridge University Press, 1995.

[10] R. A. Horn and C. R. Johnson. **Matrix Analysis**. Cambridge University Press, 1993.

[11] A. Kondratyev, M. Kishinevsky, and A. V. Yakovlev. **On Hazard-Free Implementation of Speed-Independent Circuits**. In *Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC'95)*, pages 241–248, 1995.

[12] P. Kudva and V. Akella. **A Technique for Estimating Power in Asynchronous Circuits**. In *Proc. 1st International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC), Salt Lake City, Utah*, pages 166–175. IEEE Computer Society Press, November 1994.

[13] T. Murata. **Petri Nets: Properties, Analysis and Applications**. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[14] F. N. Najm. **A Survey of Power Estimation Techniques in VLSI Circuits**. *IEEE Transactions on VLSI Systems*, 2(4):446–455, December 1994.

[15] W. Nebel. **Power Estimation at the Logic Level**, August 1996. Low Power Design in Deep Submicron Electronics - A NATO ASI, Italy.

[16] R. Parikh. **On Context-Free Languages**. *Journal of the ACM*, 13(4):570–581, 1966.

[17] J. L. Peterson. **Petri Net Theory and the Modelling of Systems**. Prentice-Hall, 1981.

[18] J. A. Tierno and A. J. Martin. **Low-Power Asynchronous Memory Design**. In *Proc. 1st International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC), Salt Lake City, Utah*, pages 176–185. IEEE Computer Society Press, November 1994.

[19] C-Y. Tsui, M. Pedram, and A. M. Despain. **Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSM's**. In *ACM/IEEE 31st Design Automation Conference*, pages 18–23, San Diego, C. A., June 1994.

[20] A. V. Yakovlev. **On Limitations and Extensions of STG model for Designing Asynchronous Control Circuits**. In *Proc. International Conf. Computer Design (ICCD)*, pages 396–400. IEEE Computer Society Press, October 1992.