Estimation of Energy Consumption in Speed-Independent Control Circuits

Peter A. Beerel and Cheng-Ta Hsieh and Suhrid Wadekar
EE-Systems Department
University of Southern California
Los Angeles, CA 90089

Abstract: We describe a technique to estimate the energy consumed by speed-independent asynchronous (clock-less) control circuits. Because speed-independent circuits are hazard-free under all possible combinations of gate delays, we prove that an accurate estimate of their energy consumption is independent of relative component gate delays and can be determined by simulating only a small number of input patterns proportional to the size of the circuit's Signal Transition Graph (STG) specification. Specifically, we calculate the *average energy per external signal transition* consumed by a circuit. This can be used to compare the energy consumption between two different circuit implementations of the same specification, to calculate average energy for a given high-level operation, and to provide average circuit power when combined with delay information.

1 Introduction

The lack of a global clock in asynchronous circuits creates circuit and architectural characteristics which motivate our approach to estimating their energy consumption:

- **Control circuits drive registers:** Because asynchronous control circuits are responsible for controlling and driving all register enable lines (in lieu of a global clock), certain outputs will have large capacitive loads. This implies that these control circuits may consume a larger percentage of total chip energy as compared to synchronous control circuits. Moreover, since capacitive loads within an asynchronous circuit can vary significantly, accurate switching estimates for each gate may be more important in asynchronous circuits than in synchronous ones.

- **Distributed control:** Control circuits in asynchronous systems tend to be more numerous and on average smaller than those in synchronous systems. In our experience, most practical specifications describe on the order of 100 external signal transitions and circuit sizes are usually less than 1K gates. Consequently, it is computationally feasible to use a more accurate power-estimation procedure that has higher complexity than procedures used for synchronous circuits.

Many synchronous switching-estimation procedures (e.g., [9, 14, 11]) assume given switching probabilities on inputs that are symbolically propagated through all circuit nodes to the outputs. This methodology has low computation complexity, which is important since many synchronous circuits can consist of greater than 10K gates. However, it is challenging for these techniques to provide accurate switching estimates on individual gates; they are much better at providing good total energy estimates. This suggests that for asynchronous circuits, naive probabilistic approaches may have lower accuracy. More importantly, it is not obvious how to quantify energy consumption in asynchronous circuits. The commonly used figure of merit *energy consumption per clock cycle* is not applicable because there is no clock.

As an alternative, Kudva *et al.* [7] suggest determining the *energy per operation* (cycle of activity) of an asynchronous circuit. They applied this approach to asynchronous circuits that use a two-phase signaling protocol for request/acknowledge handshaking. The control circuits addressed in [7], however, are assumed to be among a small set of pre-defined macro gates whose energy consumption *per output transition* is pre-calculated using SPICE. The energy per output transition for all control circuits is then incorporated into the overall estimate for energy and power consumption [7].

This paper, on the other hand, considers the problem of measuring energy consumption of a more general class of control circuits to support energy and power estimation of other types of asynchronous circuits. Specifically, we consider control circuits that are netlists of gates rather than pre-defined macro gates. These circuits are speed-independent, *i.e.*, they are hazard-free under any set of component gate delays. Moreover, the circuits can exhibit input choice, in which the environment can non-deterministically decide what external inputs to change, thereby driving the circuit into different *modes* of operation, such as a read/write mode in a memory interface controller.

Since different internal signals may change in different modes, each mode may have a different energy consumption. To obtain a unique figure of merit, it is necessary to combine the energy consumption in each of the different modes of operation into a single quantity. To do this, we assume the availability of input-choice statistics, which describe the conditional probabilities of different environmental choices. Since asynchronous circuits should be optimized for average-case delay, these statistics should also be used to guide the architecture development, logic optimization, and technology mapping. In practice, these input-choice statistics may be given by the user or estimated through behavioral simulation.

Our procedure assumes the control circuit is specified using a *Signal Transition Graph (STG)*. Starting with the STG, we derive a *Sequential Signal Transition Graph (SSTG)* that defines a small set of input sequences that are simulated to obtain the switching activity of circuit nodes. The SSTG also defines a Markov chain which under reasonable circuit assumptions is irreducible, and recurrent (see section 4). Using Markov chain analysis, we obtain a small system of linear equations whose solution, when combined with the simulation results of the SSTG, yields the average energy per external signal transition.

The organization of the paper is as follows. Section 2 gives a more complete introduction to calculating energy in asynchronous circuits. Section 3 describes the classes of circuits and specifications considered in this paper. Section 4 presents a proof that the order of concurrent transitions does not affect the energy consumption of the circuit. Section 5 describes our SSTG. Section 6 describes how the SSTG defines a Markov chain which leads to a system of linear equations whose solution is used to calculate the average energy. Section 7 describes how this work relates to research on the energy consumption of synchronous circuits and efforts to mitigate the state explosion problem in formal verification. Section 8 describes some preliminary results and conclusions.

2 Calculating energy consumption

The lack of a global clock in an asynchronous circuit and the concurrent operation of the circuit and its environment obscures any obvious time frame to measure energy consumption. If the circuit repeatedly performed the same operation, such as asynchronous datapaths, then energy per operation would be a useful measure of energy consumption [7].

Control circuits, however, do not perform a well-defined single operation repeatedly and hence energy per operation is not suitable for quantifying energy use. Kudva *et al.* [7] suggest measuring the *average energy per output transition* using SPICE simulation on the final layout, which is practical when the circuits are single pre-defined gates. Yet, for other asynchronous design styles the control circuits are synthesized automatically or semi-automatically using basic gates (*e.g.,* [2, 10]). SPICE simulation is impractical for these circuits because energy consumption estimates are desired before their physical design. Moreover, because energy might be expended in response to input signal changes prior to any output signal change, we estimate a slightly different figure of merit: average *energy per external (input and output) signal transition.*

To illustrate the issues involved in calculating energy per external signal transition, let us consider the speed-independent control circuit and its STG specification depicted in Figure 1. This control circuit, called *sbuf-send-ctl*, has 5 output signals, 3 input signals, and 9 internal signals. The arrows in the STG specification describe sequencing requirements between transitions of input and output signals. For example, the arrow between *rejpkt+/1* and *y1+/1* means that after the environment raises *rejpkt*, the circuit should raise *y1*. The two arrows into *rejpkt-* from *idlebar+* and *latchaddr+/1* respectively mean that once both *idlebar+* and *latchaddr+/1* occur, the environment can lower *rejpkt*. In addition, the STG specifies environmental choice using arrows that emanate from circles, called *places*. For example, in the above STG, there exists two arrows emanating from a place to *acksend+/1* and *rejpkt+/2* with an arrow from *reqsend+* leading into the place. This means that after *reqsend+* occurs, the environment chooses between raising *acksend* or *rejpkt*, and thereby driving the circuit into different cycles of activity. The dark dot on the transition between *idlebar-* and *rejpkt+/1* denotes the initial state of the system; the first transition that can fire is *rejpkt+/1.* (An STG's formal semantics is given in Section 3.1.)

To define energy per external signal transition for such a circuit, the following definitions are needed: a *trace* T of the circuit is a sequence of STG transitions $\langle t_1, \ldots, t_n \rangle$; $\mathcal{T}$ is the set of all traces that start at the initial state and are specified to occur by the STG; and, $\mathcal{T}^k$ is a set of all traces in $\mathcal{T}$ that have k external signal transitions. For example, in the *sbuf-send-ctl* STG, one trace in $\mathcal{T}^5$ is $T_1 = \langle$*rejpkt+/1, y1+/1, idlebar+, latchaddr+/1, rejpkt-/1*$\rangle$.

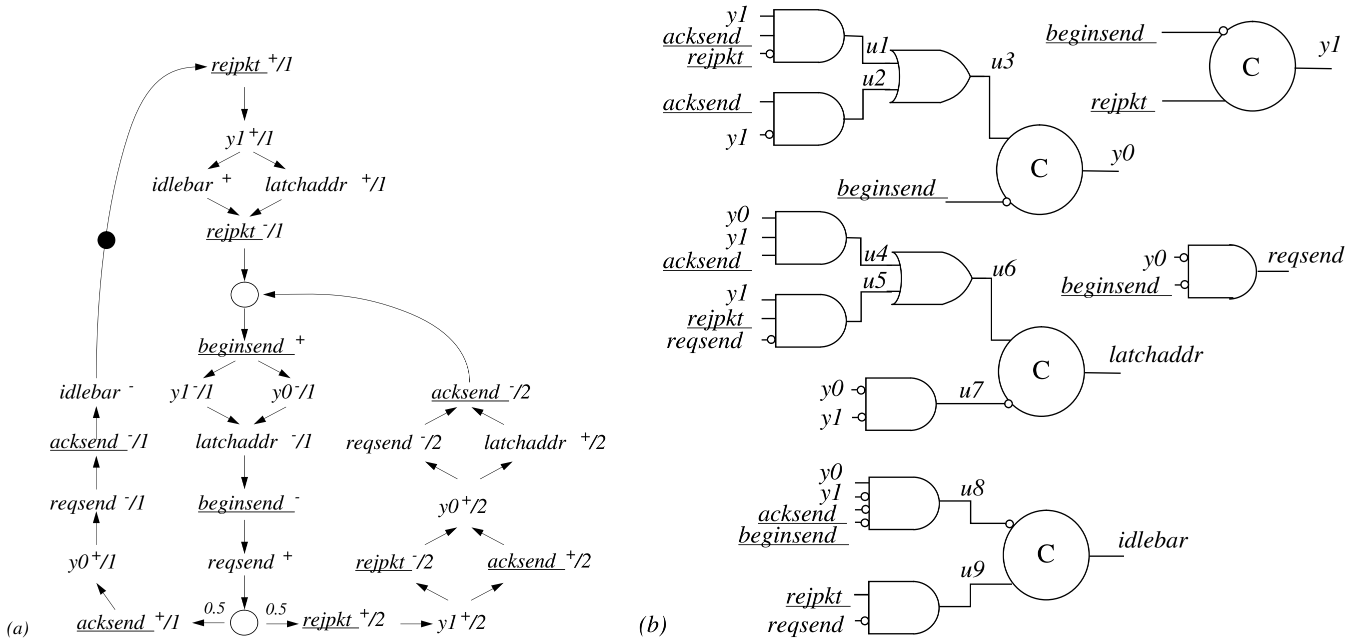

Figure 1: a) An STG specification of the *sbuf-send-ctl* circuit and b) a corresponding speed-independent implementation.

Note that there may be two signals in the STG specified to change concurrently. For example, the above specification allows *idlebar+* and *latchaddr+/1* to fire concurrently after *y1+/1* fires. In this case, each ordering of these two transitions represents one *interleaving* of the two concurrent transitions. For each interleaving there exists a different trace in $\mathcal{T}$. Hence, $\mathcal{T}^5$ also contains the trace $T_2 = \langle$*rejpkt+/1, y1+/1, latchaddr+/1, idlebar+, rejpkt-/1* $\rangle$.

Second, we define the energy consumed by a circuit during the execution of a trace T, denote $En(T)$. A trace describes only the behavior of the external circuit signals, while the energy consumed during the execution of a trace is also dependent on the behavior of the internal signals in the circuit. The number of times gates switch during execution, the order of arrival of input changes to a gate, the relative directions of the changes, and the capacitive load on a gate output all determine the energy consumption. This energy is computed *assuming the internal signals are allowed to settle before and after the execution of the trace*. For simplicity, we assume this energy depends only on the capacitance at the output of the gates that switch during the execution of T. The well-known formula for energy consumption is:

$$En(T) = \sum_{\text{gates}} \frac{1}{2} \cdot C_{\text{gate-load}} \cdot Vdd^2 \cdot (\#\text{ of gate switches}). \qquad (1)$$

A more accurate model of $En(T)$ would consider the energy consumed through charging/discharging of capacitance that is internal to gates in the circuit along with other factors, such as short-circuit current.

Consider the simulation of the trace T_1. Initially, the inputs and outputs are held fixed at the circuit's initial state and the internal signals are allowed to settle. Since, in the initial state, only signal *y0* is high, the only internal signal high is *u8*. Starting from this state, the trace is simulated and the switching activity of circuit signals is recorded. In this case, signals *y1* and *idlebar* go high, signal *u8* goes low, and signals *u5*, *u6*, and *u9* first go high and then go low. Each circuit signal transition is guaranteed to be monotonic because the circuit is hazard-free [1, 2].

Third, we associate a probability for each trace in $T \in \mathcal{T}^k$, denoted $Pr_{STG}(T)$, subject to the constraint that $\sum_{T \in \mathcal{T}^k} Pr_{STG}(T) = 1$ for all k. This probability depends on gate delays and the probability of various environmental choices. For example, consider the two traces T_1 and T_2 in $\mathcal{T}^5$. Since they are the only two traces of length 5, the sum of their probabilities should equal 1. Their exact probabilities, however, depend on the relative delays of the sub-circuits that control them. For example, since the sub-circuit for *latchaddr* has 3 levels of logic and the sub-circuit for *idlebar* only has 2 levels, *idlebar+* will most likely fire first. Since *idlebar+* fires first in T_1, a delay analysis might generate a probability of 0.8 for T_1 and 0.2 for T_2.

Finally, we can present our definition for average energy per external signal transition, denoted $\overline{E}$ as follows:

$$\overline{E} = \lim_{k \to \infty} \frac{\sum_{T \in \mathcal{T}^k} En(T) \cdot Pr_{STG}(T)}{k}. \tag{2}$$

The focus of the paper is to provide an efficient technique to calculate the above equation for a large class of speed-independent control circuits. To do this, we prove that for the class of speed-independent circuits we consider, any pair of traces T and T' that differ only in the order of concurrent transitions consume the same energy, *i.e.*, $En(T) = En(T')$. For example, the traces T_1 and T_2 discussed in the last section differ only in the order of concurrent transitions. Hence, both traces consume the same amount of energy.

As a result of this property, an *equivalence relation* that identifies traces that consume equal energy, differing only by the ordering of concurrent transitions, can be defined. This relation partitions the traces $\mathcal{T}$ into a set of *equivalence classes* Φ. We let Φ^k represent the set of equivalence classes of the traces of length k. Each class $[T] \in \Phi^k$ contains all traces of length k in which the same input choices are made by the environment as in trace T. We define the probability of an equivalence class $Pr_{STG}([T])$ to be the sum of the probabilities of all traces inside the class. The energy consumed by an equivalence class, $En([T])$, is equal to the energy consumed by any trace T in $[T]$. Average energy can then be computed based on the probability of each equivalence class $Pr_{STG}([T])$ as follows:

$$\overline{E} = \lim_{k \to \infty} \frac{\sum_{[T] \in \Phi^k} En([T]) \cdot Pr_{STG}([T])}{k}. \tag{3}$$

This result, proved in Section 4, is the key justification of our power estimation procedure. This equation refers to the probability of equivalence classes rather than that of individual traces. In the presence of concurrent transitions, the probability of individual traces can depend on relative delays inside the circuit and the environment. In contrast, the probability of an equivalence class depends only on the statistics of various input choices made by the environment. Hence, this result enables us to estimate energy consumption without performing a complicated delay analysis on the circuit or its environment.

Moreover, this equation enables us to calculate average energy by simulating the circuit using a derived sequential signal transition graph that defines a subset of the traces of the STG and does not model any concurrency. We will show that because of the lack of concurrency, the SSTG defines a Markov chain. Using Markov chain analysis, the proportion of each STG transitions that fires in an average trace in the SSTG can be calculated. This result leads to a small system of linear equations that can be used to calculate $\overline{E}$.

3 Specifications and implementations

In this paper we restrict ourselves to free-choice STG specifications and their speed-independent implementations.

3.1 Free-choice STG

An STG [3] is a specification formalism for asynchronous sequential circuits. It is an interpreted Petri net, and as such, can explicitly capture *causality*, *concurrency*, and *choice*.

A Petri net is a triple $N = \langle P, R, F \rangle$, where P is a set of *places*, R is a set of *transitions*, and $F \subseteq (P \times R) \cup (R \times P)$ is the *flow relation*. A place $p \in P$ is a *predecessor* of a transition $t \in R$, and t is a *successor* of p if $(p, t) \in F$. Conversely, a transition $t \in R$ is a predecessor of a place $p \in P$, and p is a successor of t if $(t, p) \in F$.

A *free-choice net* (FC net) is a Petri net where if a place p has more than one transition as its successors, then p must be the only predecessor of its successor transitions. Such a p is called a *free-choice* place.

An STG is an *interpreted free-choice Petri net*: transitions of the net are interpreted as value changes on external signals of the specified circuit. *Positive* transitions (labeled with a "+") represent $0 \to 1$ changes, *negative* transitions (labeled with a "-") represent $1 \to 0$ changes. *Input transitions*, which are underlined for emphasis in Figure 1, are those that occur on circuit input signals I, *output transitions* are those that occur on circuit output signals O.

The conventional graphical representation of an STG is slightly different than conventional Petri net representations. STGs are represented using a directed graph, where transitions are identified by their name, places are denoted by circles, and directed edges represent elements of the flow relation. Places with only one predecessor and one successor are omitted. Directed edges whose successor is a transition represent sequencing constraints, either on the circuit to be synthesized (if their successor is an output transition), or on the environment (if their successor is an input transition). They specify what set of transitions cause each transition.

A *token marking* of a Petri net is a non-negative integer labeling of its places. A transition is *enabled* (*i.e.*, the corresponding event can happen in the circuit) whenever all its predecessor places are marked with at least one token.

An enabled transition may *fire*. This means that the corresponding external signal changes value in the circuit. When it fires, a token is removed from every predecessor place, and a token is added to every successor place.

If a place marked with only one token has more than one enabled successor transition, then only one of them may fire non-deterministically. The other transitions are *disabled* by its firing. Hence, such successor transitions are referred to as *choice transitions.*

A marking M'' is *reachable* from another marking M' if there exists a sequence of enabled transition firings that produces M'' starting from M'. A marking M' is *live* if for all markings M'' reachable from M', every transition can be enabled through some sequence of firings from M''. A marked net is live if its initial marking is live. A marking M' is bounded if the number of tokens that any place can be holding after any sequence of firings from M' is bounded. A marking M' is *safe* (sometimes referred to as 1-bounded) if its is 1-bounded. A marked net is 1-bounded (safe) if its initial marking is 1-bounded (safe).

Two transitions are said to be *concurrent* iff there exists a marking, reachable from the initial marking, in which both transitions are enabled and from which the firing of one transition does not disable the other.

An STG is live iff 1) the underlying free choice net is live and safe, 2) no two transitions of the same signal are concurrent, and 3) if two up-transitions of a signal are fired, a down-transition of the same signal should always be fired in between (and vice versa). In this paper, we restrict ourselves to live STGs.

We annotate choice transitions with choice probabilities that designate the probability of the environment firing one of multiple mutually-exclusive input transitions. In our example, we annotated both choice transitions *acksend+/1* and *rejpkt+/2* with a probability of 0.5.

The reachability graph of a Petri net is a directed graph where each node corresponds to a marking and an edge joins a pair of marking M' and M'' if there exists a transition t that firing from M' produced M'' (the transition labels the edge).

The *state graph* (SG) [3] of an STG is the reachability graph of the underlying net where each node (henceforth called *state*) is labeled with a vector of signal values. For a given state s, the value of signal u is given by $s(u)$. This node labeling must be consistent with the SG edge labeling, in other words for each edge $s \rightarrow s'$ and each signal u: if the edge is labeled u^+, then $s(u)$ must be 0 and $s'(u)$ must be 1; if the edge is labeled u^-, then signal $s(u)$ must be 1 and $s'(u)$ must be 0; and otherwise $s(u)$ must equal $s'(u)$. The labeling can always be done, as proven in [3], if the STG is live. The procedure to find a SG from a STG is called *token flow* and is given in [3]. A portion of the SG along with the corresponding portion of the STG is depicted in Figure 2.

Notice that the SG can be exponentially larger than its STG. The size difference is dependent on the amount of concurrency expressed in the STG. In the STG, concurrency is expressed implicitly with independent parallel paths of transitions, while, in a SG, each different interleaving is explicitly represented by different paths through the SG. The two different interleaving T_1 and T_2 that were discussed earlier correspond to the two paths through the portion of the SG depicted in Figure 2.

Our procedure for estimating power does not explore the various interleavings and hence avoids the exponential complexity associated with the state graph.

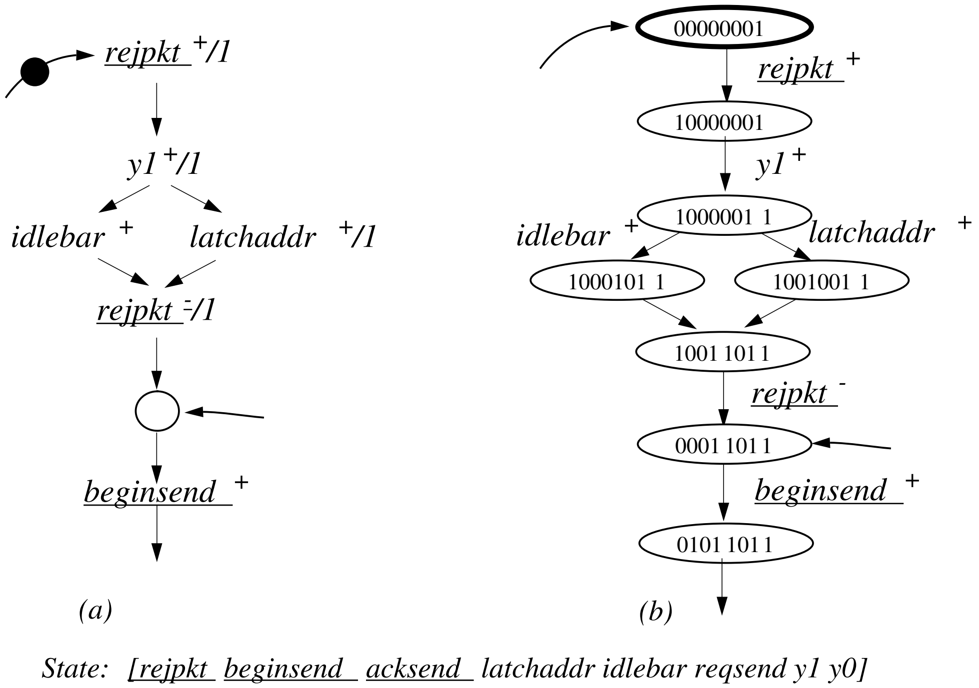

Figure 2: A portion of the STG and SG for *sbuf-send-ctl*.

3.2 Speed-independent circuits

We model a circuit with a five-tuple $G = \langle I, O, N, E, F \rangle$. The sets I, O, and N are the input, output and *internal* signals of the circuit respectively. They are collectively called the *circuit signals* and denoted A_{Impl}. $E \subseteq A_{Impl} \times A_{Impl}$ is a set of directed edges that defines the connectivity of the circuit signals. An edge $e = (u, v)$ means that signal u is a fanin of signal v. F is a labeling function that labels each internal and output signal u with a binary function f_u that describes the function of the gate that drives this signal. Two-output gates such as mutual exclusion elements cannot be modeled in this framework. Consequently, circuits that contain such gates are not considered.

To model circuit behavior we introduce a Cumulative Diagram (CD) [12, 8] which describes the history of execution of the circuit. In particular, the cumulative diagram stores the number of changes on each signal in the circuit (primary input, primary output, *and* internal). Each node in the diagram c is a *cumulative state* which represents an equivalence class among possible sequences of transitions that are simple permutations of each other. We let $[\alpha]$ denote the equivalence class for the possible sequence of transitions α. It is convenient to represent $[\alpha]$ with an non-negative integer vector c of length A_{Impl}, where $c(u)$ denotes the number of times signal u has changed in α for each $u \in A_{Impl}$.

Figure 3 depicts a simple speed-independent circuit and the initial fragment of its CD. Consider the sequence of signal transitions α equal to a^+ followed by b^+. This represents the cumulative state [110], which should be interpreted as a and b change once in this sequence. The sequence b^+ followed by a^+ is also in the equivalence class $[\alpha]$. It is important to realize that a CD models sequences of transitions that can explicitly include internal signals as well as external signals. When we say "sequence of transitions" we can mean sequences of external signals or sequences of all circuit signals (including internal signals). We hope our intended meaning can be derived from the context of the usage.

In each cumulative state c it is easy to find the corresponding logical value of a circuit signal u after any trace that leads to this state. Lavagno proved this value is unique [8]. For example, assume that the u is initially low. If $c(u)$ is even, then u is after such a trace is low, otherwise it is high. An internal or output signal is *enabled* in a cumulative state c if its logical value does not equal the value of its function given the logical values of its fanin in c. For example, if the current value of u is 0 and it is driven by an AND gate whose inputs are all 1, then u is enabled. A primary input, on the other hand, is *enabled* in a cumulative state c when the specification dictates (the formal definition is outside the scope of this paper). When an enabled signal u in a cumulative state c changes value, the signal is said to *fire*. This corresponds to moving to a new cumulative state in the CD that is formed from the old state by incrementing $c(u)$.

An internal or output signal is *disabled* if, before it fires, one or more of its fanins fire such that the circuit enters a cumulative state in which it is no longer enabled. In practice, the disabling of an internal or output signal can manifest itself as a voltage spike or runt pulse. For a broad class of circuits using a conservative delay model, there always exists a set of delays such that this spike can propagate to a primary output [1]. Therefore, such a disabling is generally considered a *hazard* and should be avoided. In addition, we assume that in response to every external signal transition, only a finite number of internal signal switchings can occur before the circuit settles. This assumption ensures that our simulation of any trace will eventually complete and is true in essentially all practical circuits. For the sequel, a circuit is *speed-independent* iff the circuit satisfies this assumption, is guaranteed to be void of hazards under all possible gate delays (assuming negligible wire delay), and satisfies its specification [1, 8].

4 Concurrency and energy consumption

This section first proves that two traces that differ only in the order of two concurrent transitions consume the same energy. Then, it extends the result to all members of an equivalence class of interleavings. The proof requires a review of some lattice theoretic properties of the cumulative diagram of speed-independent circuits.

The CD of any circuit defines a partial order over cumulative states. This order is a subset of the natural integer vector ordering and is built upon the prefix order between sequences of transitions of circuit signals up to a permutation. Formally, $[\alpha] \sqsupseteq [\beta]$ if there exists $\alpha \in [\alpha]$ and $\beta \in [\beta]$ such that β is a prefix of α.

Let C be the partial order over cumulative states C. An element $c_{lub} \in C$ is a *least upper bound* (l.u.b.) of two elements $c_1, c_2 \in C$, if $c_{lub} \sqsupseteq c_1, c_{lub} \sqsupseteq c_2$, and $c' \sqsupseteq c_1 \ \wedge \ c' \sqsupseteq c_2 \ \Rightarrow \ c' \sqsupseteq c_{lub}$. Intuitively, the least upper bound of c_1 and c_2 is the smallest cumulative state corresponding to a trace which has at least as many transitions of each signal as the maximum of c_1 and c_2. For example, in Figure 3, the least upper bound of [010] and [100] is

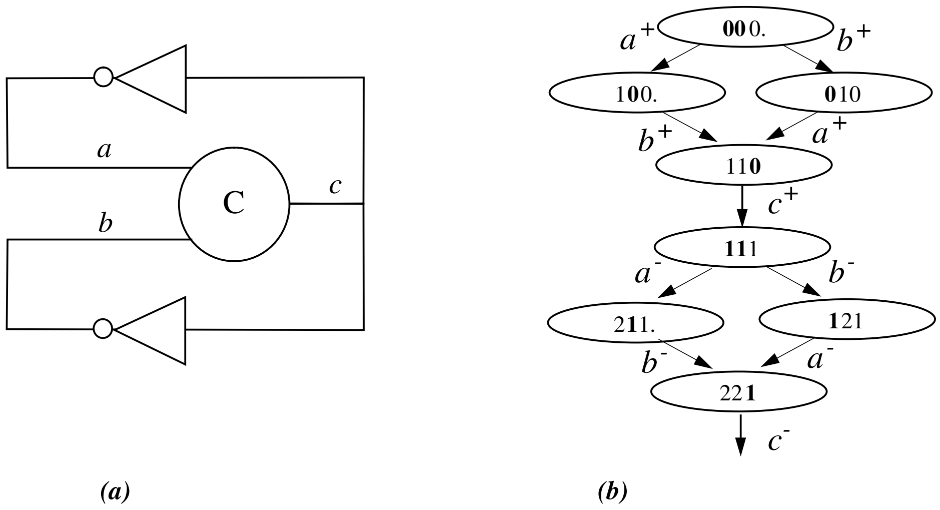

Figure 3: a) A simple asynchronous circuit and b) an initial fragment of its CD.

[110].

We can now present the key property of speed-independent circuits that is relevant to energy consumption.

Lemma 1 *Let c and c' be two cumulative states in the CD of a speed-independent circuit. The cumulative state c_{lub} is the unique l.u.b. of c and c' in which $c_{lub}(u) = \max(c(u), c'(u))$ for all signals $u \in A_{Impl}$.*

This lemma essentially states that concurrent sequences of transitions cannot disable each other, *i.e.*, after firing one sequence, the remainder of the sequence that has not been executed can subsequently fire. The rationale for this result is based on hazard-freedom. If some pending signal cannot fire, then there must exist a sequence of transitions in which that signal was disabled, violating speed-independence. The proof of this lemma is given in [8].

Using this lemma, we can now prove that simulations of traces differing only in the order of concurrent external signal transitions must arrive at the same cumulative state and thus consume the same energy. Let a binary relation $W \subseteq \mathcal{T} \times N$ be the set of pairs (T, i) such that in trace T transition t_i and t_{i+1} are concurrent. Let the function $swap(T, i)$ return the trace obtained from T by swapping the i^{th} and $i + 1^{\text{st}}$ transitions. Then, $swap(T, i)$ is *legal* if $(T, i) \in W$.

Theorem 1 *If $swap(T, i)$ is legal, then $En(T) = En(swap(T, i))$.*

Proof: (By contradiction) Let c be the cumulative state entered after executing T. (Recall that by construction after executing T we let all internal nodes stabilize.) Let $T' = swap(T, i)$. Since $swap(T, i)$ is legal, there must exist a cumulative state c' corresponding to executing T' and letting all internal nodes stabilize.

For the purpose of reaching a contradiction, presume the energy consumed in the two traces is different. Then, there exists a gate u which fires a different number of times in the execution of the two traces. This means c and

c' must be different cumulative states. Assume without loss of generality that part of the difference is that u fires less times in c than in c', *i.e.*, $c(u) < c'(u)$.

By Lemma 1, the least upper bound c_{lub} of c and c' exists and satisfies $c_{lub}(v) = \max(c(v), c'(v))$ for all $v \in A_{Impl}$. This means that there must exist a sequence of transitions α such that u fires from c. In fact, since during T and T' the exactly the same external signals fire, $c(v) = c'(v) = c_{lub}(v)$ for all external signals v. Thus, α must include only internal signal transitions. By construction of T and T', however, all internal signals are stable, so none can be enabled in c. Therefore, no sequence of internal signal transitions from α can exist. A contradiction.

We conclude that our assumption must be wrong and c and c' must be the same cumulative state. Since energy is dependent only on the number of switchings of each circuit signal, the energy consumed by T and T' must be identical. ∎

The rationale behind this theorem can also be viewed a different way. Assume there exists is an internal signal u that fires one (or more) times more in the execution of trace T' than in the execution of the trace T. First, there exists a path of cumulative states in which u is actively enabled one more time than it can fire in T. Then, by Lemma 1, there exists a second path of cumulative states, leading from the tail of the first path, that completes the simulation of T without firing u an additional time. In the final cumulative state entered, all internal signals, including u, are by definition stable. This means that u must be disabled somewhere along this second path, violating hazard-freedom.

To extend this result to any set of transitions that differ only in the interleavings of concurrent transitions, we define a relation $C \subseteq \mathcal{T} \times \mathcal{T}$ such that $(T, T') \in C$ iff there exists a sequence (including the zero length sequence) of legal swaps that transform T into T'. Then, any two traces in C consume the same amount of energy. Moreover, it is easy to prove that C is an equivalence relation. Therefore, it partitions a trace set $\mathcal{T}$ into a set of equivalence classes $[T] \in \Phi$. In particular, it partitions all traces of length k, $\mathcal{T}^k$, into the set of equivalence classes Φ^k.

The equation for average energy can be recast in terms of these equivalence classes rather than individual traces to obtain Equation 3, repeated here for convenience:

$$\overline{E} = \lim_{k \to \infty} \frac{\sum_{[T] \in \Phi^k} En([T]) \cdot Pr_{STG}([T])}{k}. \tag{4}$$

The advantage of this alternative representation is that the probabilities of each equivalence class depend only on input-choice statistics. Hence, a delay analysis of the circuit to determine relative probabilities of different interleavings caused by concurrent output signals is unnecessary.

5 The sequential signal transition graph

The energy of an equivalence class, $En([T])$, is measured by simulating a *characteristic trace* T of $[T]$ This characteristic trace is defined by imposing an order on concurrent STG transitions which is the same for all equivalence classes. Such an ordering ensures that equivalence classes that contain different subtraces corresponding to the

Algorithm 5.1 (Find SSTG)
/ Let $N = \langle P, R, F \rangle$, be the specification STG. */*
/ Let M be a marking of the STG. */*
/ Let t be a transition that drives the circuit into M. */*
/ Let fire(M, t') be the marking obtained from M*
*after firing enabled transition t'. */*
/ Let set S contain transition-marking pairs (t, M). */*
/ Let i be a time stamp. */*
Find_SSTG(N, M) {
  initialize S to include $(null, M)$ and set $i = 0$
  while *S is not empty*
    remove some transition-marking pair (t, M) from S.
    call add_state(t) to add a node for t to the SSTG
    if *there exists an unseen non-choice transition t'*
        enabled in M
      call add_edge(t, t') to connect nodes for t and t'
      mark t' as seen at time i and increment i
      add the transition-marking pair $(t', fire(M, t'))$ to S
    else if *there exists a set D of unseen mutually-exclusive*
        choice transitions enabled in M
      foreach *choice transition t' in D*
        call add_edge(t, t') to connect nodes for t and t'
        mark t' as seen at time i and increment i
        add the transition-marking pair $(t', fire(M, t'))$ to S
    else if *there exists seen transitions enabled in M*
      let t' be the seen transition with the lowest time stamp
      call add_edge(t, t') to connect nodes for t and t'
  call add_places() to add free-choice places at transitions
    with multiple predecessors or successors.
}

Figure 4: The algorithm to find the SSTG from a given STG.

same part of the STG will have the same choice of interleavings. We specify this ordering by taking the original STG and deriving a new STG, called a *sequential signal transition graph* (SSTG), in which all concurrency is serialized.

Algorithm 5.1, presented in Figure 4, finds an SSTG for a given STG. The algorithm takes as inputs the initial STG and its initial marking. Starting from the initial marking, the algorithm simulates the STG choosing only one of possibly many concurrent transitions, giving higher priority to enabled non-choice transitions over enabled choice transitions. This priority can reduce the size of the resulting SSTG. If a choice transition is chosen ahead of a concurrent non-choice transition, the non-choice transition will appear multiple times, one corresponding to each choice. With our priority scheme, the non-choice transition appears only once, before the choice is made. The SSTG is created during simulation on the fly. For each transition t visited, a node is added to the SSTG with the function *add_state*(t). (Note, *add_state*(null) is a NOP). For each transition t' fired, an edge is added to the STG with the function *add_edge*(t, t'). (Note, *add_edge*(null, t') is a NOP). The functions *add_state*() and *add_edge*() simply record their nodes and edge arguments in a global data structure that models the SSTG as a directed graph (such as an adjacency list). When there only seen transitions left in M, an edge to a previously seen transition is formed. This edge is connected to the seen transition that was placed in the data structure earliest and thus is at the head of a sequence of transitions. This ensures that the edge added creates the largest cycle of transitions possible. Otherwise, it is possible for the earliest transition to have no transitions lead to it. The graph is then transformed to model environmental choice with free-choice places in the last step. The resulting SSTG is comparable in size with the initial STG. When run on the STG given in Figure 1, the algorithm derives the SSTG illustrated in Figure 5.

Note that each trace of the SSTG T is a characteristic trace of the equivalence class $[T]$ in the original STG. In fact, the purpose of the SSTG is that the probability of the equivalence class $[T]$, which recall is defined as the sum of $Pr_{STG}(T')$ for all traces T' in $[T]$, is equal to the probability of T in the SSTG, *i.e.*,

$$Pr_{STG}([T]) = \sum_{T' \in [T]} Pr_{STG}(T') = Pr_{SSTG}(T). \tag{5}$$

To see this result, decompose $Pr_{STG}(T')$ into the product of the conditional probability of trace T' given $[T]$ multiplied by the probability of the sequence of choice decisions (which is $Pr_{SSTG}(T)$). Since the sum of the conditional probabilities over all traces in $[T]$ is 1, the result is obtained.

Using this fact, the average energy of the circuit can be computed as follows:

$$\overline{E} = \lim_{k \to \infty} \frac{\sum_{T \in \mathcal{T}^k_{SSTG}} En(T) \cdot Pr_{SSTG}(T)}{k}, \tag{6}$$

where $\mathcal{T}_{SSTG}$ is the set of all traces in the SSTG.

6 Calculating average energy

To calculate the average energy using the SSTG we define the *long term proportion* of an SSTG transition t, denoted Π_t, as follows: It is the limit of the sum of the probability of trace T times the number of occurrences of t in T divided by the length of T over all traces in $\mathcal{T}^k_{SSTG}$ as k tends to infinity:

$$\Pi_t = \lim_{k \to \infty} \frac{\sum_{T \in \mathcal{T}^k_{SSTG}} (\# \text{ of } t \text{ in } T) \cdot Pr_{SSTG}(T)}{k}. \tag{7}$$

Then, the equation for average energy can be recast as follows:

$$\overline{E} = \sum_{t \in R} \Pi_t \cdot En(t), \tag{8}$$

where R is the set of STG transitions and $En(t)$ is the energy consumed simulating transition t. Note that each SSTG transition t has a unique corresponding SG state s in which t is enabled in which simulation begins. (This state can be obtained through a reachability analysis of the SSTG.) We begin simulation of the transition

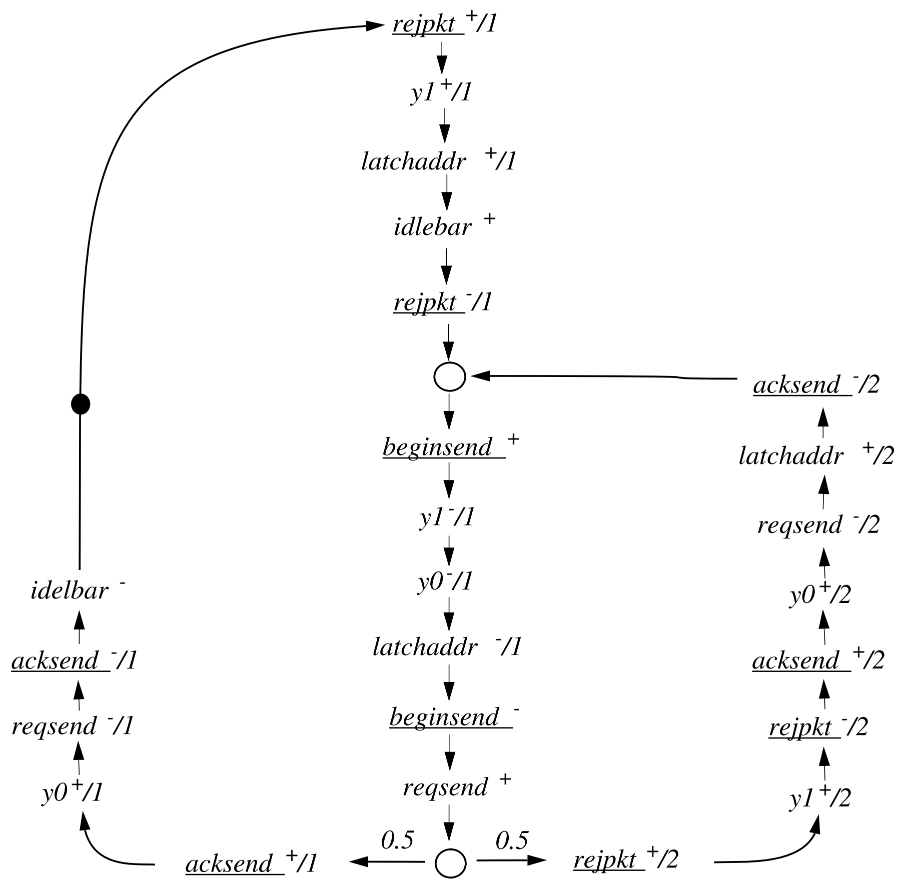

Figure 5: The SSTG for the circuit *sbuf-send-ctl*.

assuming all internal circuit signals settle in this state. Then, we fire the external signal and simulate the circuit until all internal signals have once again settled, recording which circuit signal change. Notice that our definition of speed-independence guarantees that the circuit will eventually settle. During simulation, a zero delay model can be used for circuits which have no internal feedback since internal signals can change at most once per external signal transition. Many automatically synthesized circuits have this property, *e.g.*, [2]. When internal feedback can exist, on the other hand, internal signals may change multiple times per external signal transition, requiring the use of a more realistic delay model.

Developing this equation from the SSTG is important because now *Markov chain* theory can be used to obtain the long-term proportions of STG transitions. To demonstrate this, we review Markov chains, explain why the original STG does not generally define a Markov chain, and demonstrate that, under the assumption of *uncorrelated choices*, the traces of the SSTG define a Markov chain.

Consider a stochastic process $\{X_n, n = 0, 1, 2, \ldots\}$. If $X_n = i$ then we say the process is in state i at time n. If the conditional distribution of any future state X_{n+1} is independent of the past states and depends only on the present state, then the process is a Markov chain.

First, consider the stochastic process $\{T_n, n = 0, 1, 2, \ldots\}$ that models all possible traces in the *original* STG.

In each "time step", T_n can take on a *finite* set of values, the STG transitions. If $T_n = t$, then the n^{th} state of the process is the STG transition t. The traces *do not* define a Markov chain because concurrency is allowed. Consider, for example, two concurrent transitions t and t'. The probability of a future state T_{n+1} being transition t' given that $T_n = t$ depends on whether or not t' was visited earlier (such as in state T_{n-1}), *i.e.*, the conditional distribution depends not only on the current state of the process, but also on past states. This violates the Markov model.

Now, consider the traces in the SSTG which exhibits no concurrency. The only decision points made during the trace are at choice places. If there is no correlation between consecutive input choices, the conditional distribution of the future states T_{n+1} depends only on the present state. The transition probability between states corresponding to STG transition t and t', denoted $P_{tt'}$, are as follows:

- zero if there is no edge from t to t';
- one if the only edge into t' is from t; and
- the choice probability of t, if t is a choice transition.

Hence, the stochastic process is a finite-state Markov chain. In fact, in Section 7, we will suggest that for speed-independent circuits, correlations among input choices do not affect the average energy consumption of the circuit.

Before we can present our final result, we must demonstrate the our Markov chain satisfies a few properties. A state t in a Markov chain is *recurrent* if, starting in state t, the process will ever reenter state t. Since the STG is live we are guaranteed after firing t there will exist a trace in the original STG that refires t. Since we do not allow zero choice probabilities we are guaranteed that the probability of this trace is non-zero. Hence, our Markov chain is recurrent.

A Markov chain is *irreducible* if every state can be reached from every other state. Because the original STG is live, from every marking there exists a sequence of transitions which enables every transition. Since this property extends to our SSTG, our Markov chain is irreducible.

Because our finite-state Markov chain is irredundant and recurrent, it can be shown that the Π_t are the unique non-negative solutions to the following equations [13]:

$$\Pi_{t'} = \sum_{t \in R} \Pi_t \cdot P_{tt'}, \qquad \text{for all } t' \in R \tag{9}$$

$$\sum_{t \in R} \Pi_t = 1 \tag{10}$$

Because most of the transitions in the SSTG have a single predecessor, many of the above equations are trivial. For example, since *y1+* has a single predecessor, namely *rejpkt+/1*, one of the equations is $\Pi_{y1+} = \Pi_{rejpkt+/1}$. In other words, many transitions are equiprobable and can be combined into a *sequence of transitions* L_m. The

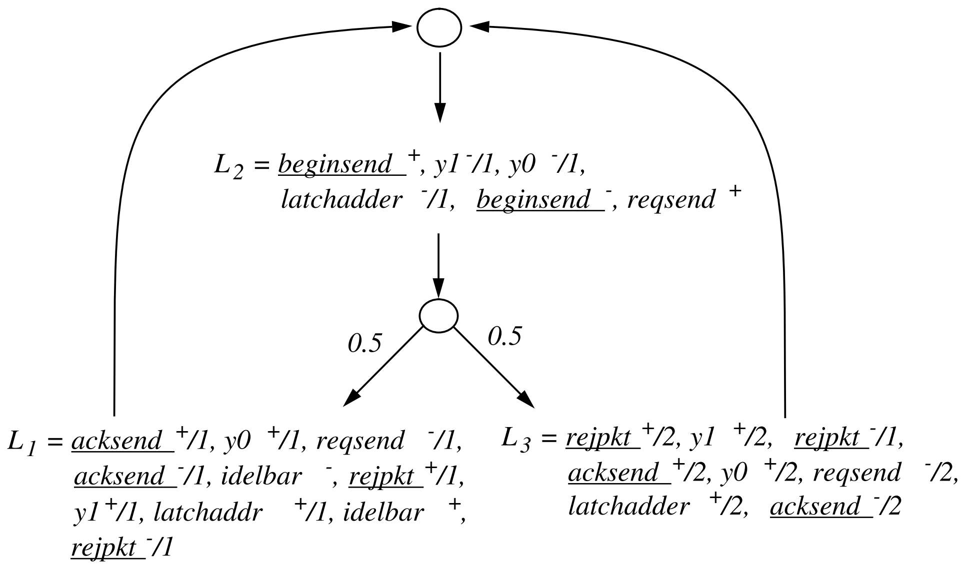

Figure 6: The collapsed SSTG for the circuit *sbuf-send-ctl*.

different sequences β form a partition of the signal transitions in the SSTG. For the SSTG depicted in Figure 5, we illustrate this collapsing of equiprobable transitions in Figure 6.

The new system of linear equations based on this partition of sequences of equiprobable transitions are as follows:

$$\Pi_{L_m} = \sum_{L_l \in \beta} \Pi_{L_l} \cdot P_{L_l L_m}, \qquad L_m \in \beta \tag{11}$$

$$\sum_{L_m \in \beta} (\#\text{ of transitions in } L_m) \cdot \Pi_{L_m} = 1. \tag{12}$$

In our example, this transformation reduces the number of needed equations from 24 to 4. Average energy can then be calculated in terms of these transition lists as follows:

$$\overline{E} = \sum_{L_m \in \beta} \Pi_{L_m} \cdot En(L_m), \tag{13}$$

where $En(L_m)$, the energy consumed in the transition list L_m, is simply the sum of the energy consumed in its composite transitions. Hence, the exact value of $\overline{E}$ can be obtained by simulating each STG transition once!

7 Relationship to related work

Because the performance of a synchronous circuit is fixed by the period of the global clock, the average circuit power can be obtained from the energy consumed per clock cycle. As a result, the *switching activity* (the expected number of times a gate switches in one clock period) of circuit signals determines the energy and power consumption estimates. The goal of current research is to obtain the most accurate estimate of switching activity while maintaining low complexity.

Many synchronous energy estimation techniques address taking into account correlations among different circuits signals. Correlations among signals can be related temporally (on the same signal at different times) [5], spatially (between different signals at the same time), and spatio-temporal correlations (different signals at different times) [9, 14, 11].

In our circuits, the only correlations that are not taken into consideration by our Markov model are correlations between different input choices. Correlations among the input choices affect the probability of each trace. More specifically, they affect the probability of sequences of transition lists. They do not affect, however, the expected number of occurrences of a particular transition list in an average trace. Since the energy consumed in a circuit depends only on the expected number of occurrences of each transition list, we conjecture that correlations do not affect average energy.

Interestingly, recent work in the area of formal verification also takes advantage of the invariance of concurrent events. In particular, researchers have proven that the choice of interleaving of concurrent events does not affect the validity of many safety properties [6, 15]. Thus, the choice can be arbitrarily made which can significantly reduce the number of reachable states needed to be analyzed [6, 15]. This parallels our proof that the various interleavings of concurrent transitions consume the same energy and the construction of the SSTG which arbitrarily identifies one of many interleaving to be analyzed.

8 Results and Conclusions

We have automated our procedure in a CAD tool and applied the tool to several circuits whose STG specifications are modified versions of those used in a large industrial circuit [4] and report the results in Table 1. Since these specification are taken from a benchmark set [8], their use in a large system is unknown. Hence, we had to make some assumptions about their use. First, we assumed that all mutually exclusive choice transitions are equiprobable. Since for most of these circuits layed out was not available, we had to make some assumptions on the load capacitances of the gates in the circuit. Second, we assume each fanout of a signal contributes 25 fF to the output load of the gate. The circuit netlist provides the fanout of internal signals while each primary output is assumed to have a total of 4 fanouts. Third, we assume a 5 volt power supply is used.

The first column shows the number of signal transitions in the original STG, the second column shows the number of circuit signals, and the third column shows the number of different transition lists obtained by serializing and then collapsing the original STG. The last column represents our final estimate for average energy per external signal transition.

In addition, to help validate the soundness of our method we performed a comparison between energy consumption per transition estimates obtained from our SSTG algorithm values obtained via SPICE simulations. The circuit *sbuf-send-ctrl* was layed out using custom gates in a 2.0 micron SPDM technology. The circuit netlist obtained from the layout was simulated for 10 runs, each consisting of 500 randomly generated input vectors,

Average Energy Estimates				
Circuit	# of STG transitions	# of circuit signals	# of transition lists	Average energy
mp-forward-pkt	16	13	1	0.98 pJ
ram-read-sbuf	22	16	1	0.85 pJ
sbuf-send-pkt2	27	17	6	0.90 pJ
sbuf-send-ctl	24	17	4	1.00 pJ
pe-rcv-ifc	50	40	9	1.43 pJ
pe-send-ifc	53	38	8	1.38 pJ

Table 1: Calculated average energy estimates assuming 25 fF gate capacitance and a 5V power supply.

using SPICE Version 3f3.[1] We generate these vectors by simulating the STG and randomly selecting which enabled transition to fire using 10 different random number generator seeds. When a choice (or place) node is encountered, the random choice was biased appropriately by the choice probability. First, hazard-free behavior was verified for the simulation by observing all the primary outputs. Then, energy consumed from the source was then computed by integrating the *Vdd* branch current over the simulation time.

The 10 simulation runs varied between giving 1.9051 pJ and 2.3051 pJ average energy per external signal transition. This variance is due to differences in the random number sequence, *i.e.*, the random simulator may choose to simulate a slightly higher (lower) percent of less energy-expensive external signal transitions in different runs. The average of the 10 different runs gives an average energy of 2.129 pJ and our calculated result using extracted capacitances is 2.012 pJ. Thus, the measured results is on average approximately 5.5% larger than the calculated result. This difference is expected since our model does not account for short-circuit current which is known to consume approximately this percentage of overall energy.

This experiment also shows the run-time advantage of our program. Each simulation run of 500 vectors on *sbuf-send-ctrl* took well over 3 hours. The total CPU time for our program to compute average energy for all of the circuits in the table (combined) is less than 1 second.

In conclusion, the main purpose of the technique proposed in this paper is to facilitate an energy consumption comparison between two implementations of the same STG. Because the complexity of the approach is very low, this technique may guide logic optimizations (such as those described in [1]) to facilitate the synthesis for low power. In addition, when combined with the average number of transitions per high-level operation, the average energy per operation can be obtained. This will enable circuits with different STG specifications for the same application to be compared. This technique is also directly applicable to bounded-delay circuits specified with signal transition graphs (such as generated by SIS [8]). However, because these circuits are not speed-independent, the results would only be an approximation of average energy. Finally, using additional techniques described in [7], we can also estimate the energy consumption per operation for asynchronous circuits that contain both datapath and control circuits.

[1] We used the netlist extractor/translator program "ext2sim Ver. 6.3", which ignores source and drain capacitances. Thus, SPICE assumes source/drain capacitance values to be 0. In order to be consistent with SPICE we made the same assumptions in our estimation algorithm.

Acknowledgements

We would like to thank Massoud Pedram, George Papavassilopoulos, and Alice Parker of the University of Southern California for their guidance and support. We'd also like to thank Chris Myers of the University of Utah, Kenneth Yun of the University of California at San Diego, and Steven Nowick of Columbia University for stimulating discussions on power estimation of asynchronous circuits. In addition, we would like to thank the anonymous reviewers for excellent constructive criticism. This research was funded in part by a Charles Lee Powell Foundation Research Equipment Grant for Young Faculty of the USC School of Engineering and a NSF Career Grant.

References

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

[2] P. A. Beerel and T. H.-Y. Meng. Automatic gate-level synthesis of speed-independent circuits. In *IEEE ICCAD* Digest of Technical Papers, pages 581–586, 1992.

[3] T.-A. Chu. *Synthesis of Self-Timed VLSI Circuits from Graph-theoretic Specifications.* PhD thesis, Massachusetts Institute of Technology, 1987.

[4] B. Coates, A. Davis, and K. Stevens. The Post Office experience: Designing a large asynchronous chip. *Integration, the VLSI journal*, 15(3):341–366, Oct. 1993.

[5] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of average switching activity in combinational and sequential circuits. In *Proc. ACM/IEEE Design Automation Conference*, pages 253–259, 1992.

[6] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. *Formal Methods in System Design*, 2:149–164, 1993.

[7] P. Kudva and V. Akella. A technique for estimating power in asynchronous circuits. In *International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 166–175, 1994.

[8] L. Lavagno. *Synthesis and Testing of Bounded Wire Delay Asynchronous Circuits from Signal Transition Graphs.* PhD thesis, University of California, Berkeley, 1992.

[9] R. Marcalescu, D. Marcalescu, and M. Pedram. Switching activiy analysis considering spatiotermporal correlations. In *Proc. International Conf. Computer-Aided Design (ICCAD)*, pages 294–299, 1994.

[10] A. J. Martin. Programming in VLSI: from communicating processes to delay-insensitive VLSI circuits. In C. Hoare, editor, *UT Year of Programming Institute on Concurrent Programming.* Addison-Wesley, 1990.

[11] J. Monteiro, S. Devadas, and B. Lin. A methodology for efficient estimation of switching activity in sequential circuits. In *Proc. ACM/IEEE Design Automation Conference*, pages 12–17, 1994.

[12] D. E. Muller and W. S. Bartky. A theory of asynchronous circuits. In *Proceedings of an International Symposium of the Theory of Switching*, pages 204–243, 1959.

[13] S. Ross. *Introduction to Probability Models.* Academic Press, 1985.

[14] C.-Y. Tsui, M. Pedram, and A. Despain. Exact and approximate methods for calculating signal and transition probabilities in FSMs. In *Proc. ACM/IEEE Design Automation Conference*, pages 18–24, 1994.

[15] A. Valmari. A stubborn attack on state explosion. In *Proceedings of the 2nd Workshop on Computer-Aided Verification. Lecture Notes in Computer Science*, pages 176–185, 1990.

List of Figures

List of Tables

Average Energy Estimates				
Circuit	# of STG transitions	# of circuit signals	# of transition lists	Average energy
mp-forward-pkt	16	13	1	0.98 pJ
ram-read-sbuf	22	16	1	0.85 pJ
sbuf-send-pkt2	27	17	6	0.90 pJ
sbuf-send-ctl	24	17	4	1.00 pJ
pe-rcv-ifc	50	40	9	1.43 pJ
pe-send-ifc	53	38	8	1.38 pJ

Algorithm 5.1 (Find SSTG)

/ Let $N = \langle P, R, F \rangle$, be the specification STG. */*
/ Let M be a marking of the STG. */*
/ Let t be a transition that drives the circuit into M. */*
/ Let fire(M, t') be the marking obtained from M*
*after firing enabled transition t'. */*
/ Let set S contain transition-marking pairs (t, M). */*
/ Let i be a time stamp. */*

```
Find_SSTG(N, M) {
  initialize S to include (null, M) and set i = 0
  while S is not empty
    remove some transition-marking pair (t, M) from S.
    call add_state(t) to add a node for t to the SSTG
    if there exists an unseen non-choice transition t'
        enabled in M
      call add_edge(t, t') to connect nodes for t and t'
      mark t' as seen at time i and increment i
      add the transition-marking pair (t', fire(M, t')) to S
    else if there exists a set D of unseen mutually-exclusive
          choice transitions enabled in M
        foreach choice transition t' in D
          call add_edge(t, t') to connect nodes for t and t'
          mark t' as seen at time i and increment i
          add the transition-marking pair (t', fire(M, t')) to S
    else if there exists seen transitions enabled in M
      let t' be the seen transition with the lowest time stamp
      call add_edge(t, t') to connect nodes for t and t'
  call add_places() to add free-choice places at transitions
      with multiple predecessors or successors.
}
```