APPENDIX A
ENERGY AND ENTROPY OF ASYNCHRNONOUS CIRCUITS

This appendix sketches a proof of a lower bound on the scalability of energy dissipated by an asynchrnonous circuit in terms of the entropy of its specification.

A.1 Circuit Specification and Entropy

Analogous to a structural or behavioral description of a synchronous system, asynchronous circuits are specified using a process algebra formalism. Any circuit description requires an *alphabet* of signals $\mathbb{C} = \{\mathbb{I}, \mathbb{S}, \mathbb{O}\}$ composed of inputs ($\mathbb{I}$), outputs ($\mathbb{O}$) and internal *state* signals ($\mathbb{S}$).

An asynchronous circuit is described as a collection of communicating *processes* – sequential programs executing simultaneously on separate machines and synchronizing by passing messages.

A *statement*, denoted by block letters except G, is a boolean assignment where a state/output signal is assigned the value of a boolean expression of any number of signals. $A; B$ is a sequential process where statement B begins only after A completes. A *guard* $(G, G_1, G_2, ...)$ is a boolean condition composed of any number of signals, used to implement control flow. There are two control structures:

- **Deterministic Selection** Denoted by $[G_1 \rightarrow S_1 \,\square\, G_2 \rightarrow S_2 \,\square\, ...\square\, G_n \rightarrow S_n]$ where at most one guard $G_1...G_n$ is true at any instant. The program waits until a guard becomes true. If G_i is true, only S_i executes.
- **Non-Deterministic Selection** Denoted by $[G_1 \rightarrow S_1 \mid G_2 \rightarrow S_2 \mid ...\mid G_n \rightarrow S_n]$. Same as Deterministic Selection, except when more than one guards are true, any one of them is selected. The selection criteria is not necessarily random and can be assumed to be demonic for the circuit correctness.

A *repetition* on a control structure is denoted by '$*[...]$': the control structure is repeatedly executed until no guards are true. $S_1||S_2$ denotes *concurrent* execution, where the statements are executed in any order and ensures *weak fairness* (i.e. any action that is enabled to execute and stays enabled will eventually execute).

Any of the guarded statements $S_1...S_n$ in the examples above could be control structures, their repetitions, sequential processes or concurrent processes themselves. Finally for message passing, a special data structure *channel* (denoted by $\mathbf{1}, \mathbf{2}...$) is used. At any instant, at most one statement shall read from or write to a channel. A read statement $\mathbf{1}?v$ reads the value from the channel and stores it in signal v. A write statement $\mathbf{1}!G$ evaluates the boolean expression G and write the value to channel $\mathbf{1}$. A read and write statement waits till channel is empty or full respectively.

We postulate that any asynchronous circuit can be completely described by a process P of the form:

$$P = *[[G_1 \rightarrow A_1; X_1 \square\, G_2 \rightarrow A_2; X_2 \square\, ...\square\, G_n \rightarrow A_n; X_n]] \tag{1}$$

where $A_1..A_n$ are either **skip** or message passing statements, $X_1...X_n$ are simple assignments where the value assigned is a boolean constant [TODO: cite paper]. This essentially means all computation by the circuit is captured by evaluation of guard conditions alone.

Let Z_i be a random variable for the i$^{\text{th}}$ statement executed by P. The probability distribution for Z_i can be calculated assuming equi-probable input values or profiling actual traces of program execution.

Definition 1 (Entropy). Let $Z = (Z_1, Z_2, ...Z_m)$ be a sequence of m discrete random variables over range S_m with joint probability mass function $\Pr(Z_1, Z_2, ...Z_m)$. The entropy of the sequence is

$$H(Z_1, Z_2, ...Z_m) = -\sum_{(z_{1...m}) \in S_m} \Pr(z_1...z_m) \log_2 \Pr(z_1...z_m) \tag{2}$$

Definition 2 (Entropy of a Process). The entropy of a process P which executes sequence of statements $Z_1, Z_2, ...$ is

$$H(P) = \lim_{m \to \infty} \sup \frac{1}{m} H(Z_1, Z_2, ...Z_m) \tag{3}$$

P has about n statements $X_{1...n}$ ($A_{1...n}$ are mostly **skips** as circuits often have a lot of signal-level parallelism), hence $Z_1, Z_2, ...$ can have values n distinct values. Therefore, their entropies are bounded $H(Z_1), ...H(Z_2) \leq \log_2 n$. Hence, the entropy of the process P exists as its a limit of the supremum of a bounded sequence:

$$\frac{1}{m} H(Z_1...Z_m) \leq \frac{1}{m}(H(Z_1) + ... + H(Z_m)) \leq \log_2 n \tag{4}$$

A.2 Energy Index of Asynchronous Circuits

A CMOS circuit has three main sources of energy dissipation: leakage currents, short circuit currents and dynamic switching currents flowing between the rails. An *operation* is defined as a *finite* sequence of input transitions within a finite time. The input values may not be provided, only the number of transitions and their times must be specified [TODO: cite trace theory]. Assuming most of the circuit is implemented using static CMOS (dynamic nodes are usually pulled to rail by weak feedback to prevent soft errors) and designed to have low leakage, we can estimate the total energy expended **in one operation** of the circuit:

$$E_T \approx E_{sw} = \sum_i n_i C_i V_{DD}^2 \tag{5}$$

Here, all nodes (indexed by i) have capacitance C_i and switch LOW to HIGH n_i times during one operation. The intermediate nodes of stacked transistors are ignored (they are usually smaller than drain capacitances and charge to values lower than V_{DD}).

It is worthwhile to note differences of this energy model to that of a synchronous circuit. The switching factor n_i is an integer rather than a probability. Also, a well-defined operation requires that outputs stabilize within finite time after the inputs switch. A special class of asynchronous circuits are required to avoid metastability [TODO: cite

book].

The *energy index* of the circuit is defined as $K = \sum_i n_i C_i$. The energy indices are additive: the index of a circuit is a sum of energy indices of all its sub-circuits. The energy index of a CSP P is denoted by $C(P)$.

A central assertion of this model is that parallel sub-circuits with no synchronization require no extra energy. This is reasonable since no synchronization essentially means no extra circuitry, only extra wires for common signals.

Since asynchronous circuits compute only when required, the chances of a sub-circuit turning on and consuming switching energy depends on the inputs. If probability of sub-circuits P_1 and P_2 turning on during one operation are w_1 and w_2, then the total energy index is $K = w_1 K_1 + w_2 K_2$.

The three possible ways of enforcing order in CSP are: sequential ordering (;), choice (deterministic and non-deterministic) and message-passing. Energy models of implementing each of these are supplied without proof, please refer [TODO: cite] for the same.

Receiving and transmitting N-bit values over a channel requires energy index proportional to N. A guarded choice $P = *[[G_1 \rightarrow P_1 \Box \ ... \ \Box \ G_n \rightarrow P_n]]$ entails an extra energy index $\propto \log_2 n$ over executing the guarded statements concurrently $Q = *[[G_1 \rightarrow A_1]] \ || \ *[[G_2 \rightarrow A_2]] \ ... \ || \ *[[G_n \rightarrow A_n]]$. This is because we can implement the guarded choice using concurrent statements and extra $\log_2 n$ single-bit channels for message-passing. [TODO: cite Phd thesis]

Now consider the program P from from eqn. 1 with $n = 2$. The energy index can be evaluated as

$$\begin{aligned} C(P) = C(G_1) + C(G_2) + k \log_2 2 + \\ w_1 C(A_1; X_1) + w_2 C(A_2; X_2). \end{aligned} \tag{6}$$

Here k is a technology constant. Notice that the energy indices of all the guards are added irrespective of their probabilities being true.

Without loss of generality, assume $w_1 \geq w_2$. Now consider the following modification to the program:

$$Q = *[[G_1 \rightarrow A_1; X_1 \Box \ [\overline{G_1} G_2 \rightarrow A_2; X_2]]] \tag{7}$$

The modified energy index is

$$\begin{aligned} C(Q) &= C(G_1) + w_1 C(A_1; X_1) + k \log_2 2 + \\ &\quad (1 - w_1)(C(G_2) + u + w_2 C(A_2; X_2)) \\ &\approx C(G_1) + (1 - w_1) C(G_2) + k \log_2 2 + \\ &\quad w_1 C(A_1; X_1) + (1 - w_1) w_2 C(A_2; X_2) \\ &< C(P) \end{aligned} \tag{8}$$

Here the constant u is the fixed energy density requires to compute a logical negation and a logical AND of the result (usually negligible). Therefore, re-structuring the program P such that a more common guard executes before a less common guard saves overall energy.

For the general case (n guarded statements), assume P is re-structured to a tree of cascaded guarded statements Q as above. Let $Y_i = A_i; X_i$ denote the guarded statements of P. Then, for a particular Y_i of P there is a corresponding order of guard evaluation through the tree in Q which ends in the leaf node for Y_i. At a depth d in the tree of Q, energy will be dissipated by *all guard evaluations* at that depth. Let $S(Y_i) \subset \{G_1, G_2, ...G_n\}$ be the set of all guards that were evaluated by Q to reach leaf node for Y_i. Further, let $d(Y_i)$ denote the depth of the leaf node Y_i in the tree of Q, and let $n_1(Y_i), n_2(Y_i), ...n_{d(Y_i)}(Y_i)$ be the number of guards that were evaluated at depth 1, 2, ... $d(Y_i)$ respectively.

Then, the energy index for evaluating Y_i is given by

$$C_{Y_i}(Q) = \sum_{G \in S(Y_i)} C(G) + \sum_{j=1}^{d(Y_i)} k \log_2 n_j(Y_i) + C(Y_i) \tag{9}$$

and the energy index for evaluating $Y_{1...m} = Y_1, Y_2, ...Y_m$ by Q is $C_{Y_{1...m}}(Q) = \sum_{i=1}^{m} C_{Y_i}(Q)$.

Definition 3 (Energy Index per Operation). $\Pr(y_1, ...y_m)$ is the probability of $(y_1, y_2, ...y_m)$ being executed by P. Then, the energy index **per operation** of Q is defined as the expected average energy index of a chain of operations

$$C(Q) = \lim_{m \to \infty} \frac{1}{m} \sum_{(y_1, ...y_m)} \Pr(y_1, ...y_m) C_{Y_{1...m}}(Q) \tag{10}$$

The above expression is really the average energy index in the limit of long chains. Assuming the statements $y_1, y_2, ...$ are independent and identically distrubuted, using the law of large numbers we have:

$$C(Q) = \mathbb{E}\Big[\sum_{G \in S(Y)} C(G) + C(Y) \Big] + k \, \mathbb{E}\Big[\sum_{j=1}^{d(Y)} \log_2 n_j(Y) \Big] \tag{11}$$

Now, the path to Y is determined by which of the guards resolved true at each depth. Since the node/guard at depth i can be described by a binary number $\log_2 n_i(Y)$ bits wide, the path can be described by the list of such numbers in total $\sum_{i=1}^{d(Y)} \log_2 n_i(Y)$ bits wide. This is a prefix code for Y, and $\mathbb{E}[\sum_{i=1}^{d(Y)} \log_2 n_i(Y)]$ is the average code-length. A result from Information Theory states that the average code length of any prefix code is atleast the entropy of the alphabet, [TODO: cite] hence $\mathbb{E}[\sum_{i=1}^{d(Y)} \log_2 n_i(Y)] \geq H(Y)$.

Therefore, we have

$$C(Q) \geq (\sum_{i=1}^{n} w_i C(Y_i) + \mathbb{E}[\sum_{G \in S(Y)} C(G)]) + kH(Y) \tag{12}$$

The first term in parenthesis is the indispensable cost for evaluating the guards and executing the statements. As such, asynchronous paradigm does not provide any major benefits.

The last term relates to the *scalability* of energy with the design size. Unlike synchronous systems where the energy scalability factor is linear (or even polynomial) in size, an optimally designed asynchronous circuit scales linearly with the *entropy* of the system. This shows the *asymptotic* utility of asynchronous circuits for energy efficiency in large systems.

In this sketch, many details about methods of specification and transformation were not provided. This theory is very thoroughly developed in [TODO:cite].