APPENDIX A
ENERGY AND ENTROPY OF ASYNCHRNONOUS CIRCUITS

A CMOS circuit has **three main sources** of energy dissipation: leakage currents, short circuit currents and dynamic switching currents 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:

$$E_T \approx E_{sw} = \sum_i n_i C_i V_{DD}^2 \tag{1}$$

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 synchrnonous 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 asynchrnonous circuits are required to guarantee avoidance of metastability [TODO: cite book].

This appendix sketches an interesting algorithmic bound on the *energy index* $K = \sum_i n_i C_i$ in terms of the entropy of the circuit specification. Please refer [TODO:cite] for a thorough development of the theory.

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. Many such formalisms have been developed [TODO: cite all]: Communicating Sequential Processes (CSP), I-Nets, Signal Transition Graphs (STG).

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 ($\mathbb{S}$). Henceforth, a signal (denoted by small letters) would mean any member of the alphabet $v \in \mathbb{C}$. I give a very brief description of a minimalist CSP which I shall use for the proof of the result.

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 symbols, 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 Choice** 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. Implemented as a queue of size one, channels are *hidden variables* used only for successful simulation of the circuit. 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 $\mathbf{1}$.

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

$$P = *[[G_1 \rightarrow A_0; X_0 \square\, G_2 \rightarrow A_1; X_1 \square\, ... \square\, G_n \rightarrow A_n; X_n]] \tag{2}$$

where $A_0 .. A_n$ 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].