Optimal frequency selection in circuit designs for energy minimization

Bruno Gaujal[1] and Eric Thierry[1]

LIP, ENS-Lyon, INRIA, CNRS, UCBL, 46 Allée d'Italie, 69007 Lyon, France,
Bruno.Gaujal@ens-lyon.fr, Eric.Thierry@ens-lyon.fr

Abstract. In this paper we present a quadratic time algorithm using dynamic programming to compute the set of N speeds that a processor should use to minimize its energy consumption while meeting real time constraints. This computation is based on the energy consumption as a function of the clock frequencies and a statistical knowledge of the likelihood of using any frequency in the range from zero to the maximal possible frequency.

keywords: Energy consumption; dynamic programming; circuit design.

1 Introduction

Energy consumption is one of the most important issue in modern digital circuit design.

Several techniques are used to lower the energy use: On a static level one can mention the reduction of leaks, or removing the redundancy of logical gates [2]. On the dynamic level one can think of minimizing the size of the memory [1] or dynamically changing the voltage over time [4]. Many papers in the real time community have been devoted to the latter issue However most of them consider the following problem: given a circuit which can use a fixed set of N frequencies, how to use them in order to minimize the energy consumption while meeting real-time constraints. For example under EDF scheduling, an optimal solution (off-line) is given in [9]. On-line algorithms are given in [3, 5, 6, 9] using various assumptions on the set of tasks. Under FPP, finding the optimal solution is NP-hard but approximable [10]. To the best of our knowledge, there are few theoretical studies on a different problem: how to first select those N sample frequencies? In [7], such a problem is addressed in order to minimize the worst case. However, this paper disregards the fact that some frequencies will be used more than others and also the fact that one can use two samples instead of one to replace one frequency. These two points will be addressed here: the frequency of utilization is taken into account in a measure μ and the replacement of the optimal theoretical frequency by the samples is done optimally.

When designing a processor with dynamically varying frequencies, it is very interesting to make a good preliminary selection of the frequencies that the processor is going to use in order to reduce the energy consumption. So far, in most cases, these choices are based on experimental studies, simulations and tests. The choice of the speeds is indeed a difficult problem because it depends on hardware compromises and unknown parameters. Also, the typical number N of different frequencies that a circuit can use is usually 2 or 3 so that an exhaustive search is considered feasible. However, for asynchronous circuits, for example, N tend to be much larger [8], as well as the range over which the frequencies can be chosen. This makes the problem more combinatorial.

In this paper, we deal with all these difficulties by assuming a statistical knowledge on some of these unknown parameters and by using a mathematical approach to model all the constraints of the problem and to remove the combinatorial nature of the problem.

Several factors may play a role in this choice. Firstly, hardware constraints have to be taken into account (in some cases, it is easier to select a set of doubling frequencies of the type $\{f, 2f, 4f, 8f, \cdots\}$). Secondly, the instantaneous consumption as a function of the current frequency has also to be taken into account.

In the following, we group together those two factors into a single "cost function" $g(f)$ which gives the cost (both in energy and hardware complexity) of using frequency f.

Finally, a statistical knowledge of the future use of the processor can also play an important role in the speed selections. One way to think about this last parameter is to consider a first (expensive) circuit built for experimental purposes with a very large number of available speeds. This experimental circuit runs over a typical application for the future cheaper circuit (which will only have a few available speeds) and the designer measures which speeds are used more often. This measurements are gathered under the form of a mathematical measure μ over $\mathbb{R}$.

Now, the selection of the speeds will be a compromise between their cost (g) and their likelihood (μ).

The paper continues as follows, Section 2.2 constructs the mathematical model of the problem, Section 3 presents a dynamic programming algorithm solving the problem in quadratic time in the discrete case as well as a derivation of bounds for the continuous case. Section 4 shows several extensions of the problem.

2 Presentation of the model

2.1 Construction of μ

We assume that through some statistical analysis one can derive the distribution of the frequencies (also called speeds in the following) that are the most desirable for the processor. In order to be as general as possible, this knowledge is given under the form of a measure μ over the interval $[0, f_{max}]$. In particular, this include the case of a discrete measure (something of the following flavor: on average, the processor will use the maximal speed 1/3 of the time and half the maximal speed 2/3 of the time). It also includes the case where a continuous density is mixed with a discrete measure.

Although this is a crucial point for the appropriateness of the method presented below, we will not discuss much further on the determination of the measure μ. As mentioned in the introduction, one can imagine that μ is constructed along the following lines. Construct a simulation of the circuit which can choose over a very large selection of speeds. Run this simulation several times over its typical real-time applications. Each application is decomposed into several phases (for DSPs, these are typically matrix inversions, matrix products,...) with different real time constraints. For each step, the simulation computes the best speed that verifies the real-time constraints of the application while minimizing the energy use, such as the one proposed in [9] for example. The last step is to monitor the simulation over a large number of runs, identify the frequencies that are used the most and capture this under a measure μ over the whole range $[0, f_{max}]$.

2.2 Statement of the problem

The instantaneous energy consumption of a processor depends on the frequency $f(t)$ at time t according to a function g which depends on the technology of the processor. This function is usually convex. The non-convex case is postponed to Section 4.2. In the following we assume that $f_{max} = 1$. This is possible with no loss of generality by time rescaling.

Now, we consider a theoretical model Γ for the processor that can use the whole continuum of speeds in the range $[0, 1]$. By construction of μ, at any point in time, this model uses speed f with probability $d\mu(f)$ and the corresponding instantaneous energy consumption is $g(f)$. Under stationary and ergodic assumptions on the behavior of Γ, the expected energy consumption over a large time period T is $cT \int_0^1 g(f) d\mu$ (here c is the normalization constant: $c \int_0^1 d\mu = 1$).

Actually, the real circuit C can only use a set of $N + 2$ fixed frequencies $0 \le f_1 \le \cdots \le f_N \le 1$. Therefore, to emulate the model Γ, each time Γ uses speed f, C uses a combination of the two speeds f_k, f_{k+1} such that $f_k \le f \le f_{k+1}$ with a time ratio α between the two speeds such that $\alpha f_k + (1 - \alpha) f_{k+1} = f$. This policy for allocating the finite set of speeds has been proved optimal in terms of energy consumption (see [3]), as long as the overhead induced by speed changes is neglected.

In some frequency (or voltage) scalable processors, such as Linux/RK CPUs [7], this is not the case because changing frequencies takes a lot time. However, in asynchronous processors [8] and even for some experimental synchronous CPUs with voltage scaling such as modified XScale BRH circuits [7], the overhead induced by speed changes is very small and can be neglected.

Therefore, the expected energy consumption of the real circuit C over a given time period T is $cT \int_0^1 h(f_1, \cdots, f_N) d\mu$, where $h(f_1, \cdots, f_N)$ is the linear interpolation of g over the points $(0, f_1 \cdots, f_N, 1)$ (see Figure 1).

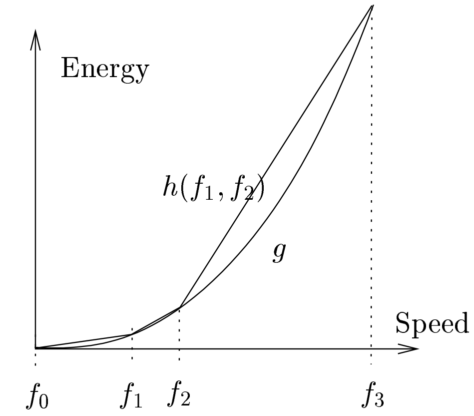

Fig. 1. The function g and its linear interpolation h.

Therefore, the problem of choosing the best frequencies is: given N, g and μ,

$$\text{Minimize } \int_0^1 h(f_1, \cdots, f_N) d\mu \text{ over all } 0 \leq f_1 \leq \cdots \leq f_N \leq 1.$$

Another way to state the problem is the following: given a convex function g, find the piecewise affine function h above g (with $N+1$ segments) closest (for the measure μ) to g.

A more explicit way to formulate this problem is by using the actual form of the function h. The problem becomes:

1 - Compute the expected energy consumption rate under $f_1 \cdots f_N$ ($H(f_1 \cdots f_N) = \int_0^1 h(f_1, \cdots, f_N) d\mu$) by decomposing the interval $[0, 1]$ into the sub-intervals $(f_k, f_{k+1}]_{k=0,\cdots,N}$ (with $f_0 = 0$ and $f_{N+1} = 1$):

$$H(f_1 \cdots f_N) = \sum_{i=0}^{N} \sigma_i,$$

with

$$\sigma_i = \begin{cases} \frac{g(f_i)f_{i+1} - g(f_{i+1})f_i}{f_{i+1} - f_i} \int_{f_i}^{f_{i+1}} d\mu + \frac{g(f_{i+1}) - g(f_i)}{f_{i+1} - f_i} \int_{f_i}^{f_{i+1}} f d\mu & \text{if } f_{i+1} \neq f_i \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

2 - Find the minimum energy consumption rate:

$$\min_{0 \leq f_1 \leq \cdots \leq f_N \leq 1} H(f_1 \cdots f_N).$$

3 Dynamic Programming Solution

Let us call $S^*(N, x, y)$ the value of the optimal expected energy consumption rate with N speeds distributed between x and y. Using the definition of $S^*(N, x, y)$,

$$\inf_{0 \leq f_1 \leq \cdots \leq f_N \leq 1} H(f_1 \cdots f_N) = S^*(N, 0, 1)$$

It is easy to see that S^* satisfies the following recurrence equation:

$$S^*(N,x,y) = \inf_{x \le z \le y} (S(0,x,z) + S^*(N-1,z,y)),$$

where

$$S(0,x,y) = \frac{1}{z-x}\left((g(x)z - g(z)x)\int_x^z d\mu + (g(z)-g(x))\int_x^z f d\mu\right).$$

This can be used to compute the optimal solution with a dynamic programming approach as soon as the state space of the speeds is finite. In order to do this, we discretize the interval $[0,1]$ with steps of length $1/k$ where k is large enough. The size of k is given in the following section (3.1).

Once this is done, we can reformulate the recurrence equation using integer variables x and y between 0 and k.

$$S^*_k(N,x,y) = \min_{z=x}^{y} (S^*_k(0,x,z) + S^*_k(N-1,z,y)),$$

where

$$S^*_k(0,x,z) = S(0,\frac{x}{k},\frac{z}{k}).$$

This provides an algorithm computing both $S^*_k(N,0,1)$ and the argmin $(f^*_1 \cdots f^*_N)$. This algorithm is given in Figure 2, where all variables have initial value 0.

```
for x from 0 to k do
    S_new[x] := S(0, x/k, 1)
for i from 1 to n do
    for x from k to 0 do
        S_old[x] := S_new[x]
        S_new[x] := ∞
        for y from x to k do
            if S(0, x/k, y/k) + S_old[y] < S_new[x] then
              S_new[x] := S(0, x/k, y/k) + S_old[y]
              f[i, x] := y
f*[n] := f[n, 0]
for i from n-1 to 1 do
    f*[i] := f[i, f*[i + 1]]
return(S_new[0] and f*[1] ··· f*[N] ).
```

Fig. 2. Dynamic Programming Algorithm computing $f^*_1 \cdots f^*_N$ with $O(k^2N)$ operations and $O(kN)$ memory space

The arithmetic complexity of the algorithm is $O(k^2N)$ operations with $O(kN)$ memory space. Using $k = p/\varepsilon$, (where p is an appropriate constant, see the next section), insures an error bound ε. This provides a quadratic complexity in the error bound and a linear complexity in the number of points.

It is also possible to modify the algorithm into a new one with $O(k^2N^2)$ operations and $O(k)$ memory space (again, all variables have an initial value equal to 0). This algorithm is given in Figure 3.

```
for x from 0 to k do
    S_new[x] := S(0, x/k, 1)
for j from n to 1 do
    for i from 1 to j do
        for x from k to 0 do
            S_old[x] := S_new[x]
            S_new[x] := ∞
            for y from x to k do
                if S(0, x/k, y/k) + S_old[y] < S_new[x] then
                    S_new[x] := S(0, x, y) + S_old[y]
                    f[x] := y
    f*[j] := f[f*[j + 1]]
return(S_new[0] and f*[1] ··· f*[N] ).
```

Fig. 3. Dynamic Programming Algorithm computing $S^*_k(N, 0, 1)$ and $f^*_1 \cdots f^*_N$ with $O(k^2N^2)$ operations and $O(k)$ memory space

3.1 Bounding k

The goal of this section is to derive the value of k given the fact that we want to approximate $S^*(N, 0, 1)$ with an error bounded by ε. The problem is to find k such that $S^*_k(N, 0, 1) - S^*(N, 0, 1) \leq \varepsilon$.

Since g is convex, the slope of g is always bounded by $p := \max(g'_+(0), g'_-(1))$.

Proposition 1. *If $k > \frac{p}{\varepsilon}$ then $0 \leq S^*_k(N, 0, 1) - S^*(N, 0, 1) \leq \varepsilon$.*

Proof. Let $f^*_1 \leq \cdots \leq f^*_N$ be the optimal solution with cost $S^*(N, 0, 1)$. For simplicity, the function $h(f^*_1 \cdots f^*_N)$ (the linear interpolation induced by $f^*_1 \cdots f^*_N$) is denoted h^*. We approximate $f^*_1 \cdots f^*_N$ by values over the discrete set $\{\frac{i}{k}\}_{i=0..k}$. We set $x_i := [kf^*_i]$[1] and $f_i := x_i/k$ for all $1 \leq i \leq N$.

Now, for simplicity, we also denote by h, the linear interpolation induced by $f_1, \cdots, f_N$, namely, $h \stackrel{\text{def}}{=} h(f_1, \cdots, f_N)$. We also construct the function w which is the piecewise affine function between the points $f^*_0, \cdots, f^*_{N+1}$ such that $w(f^*_i) := h(f^*_i)$.

[1] here, $[x]$ is the integer closest to x

We will first show that $w \geq h$. Indeed, for all $0 \leq i \leq n$, since $w(f_i^*) = h(f_i^*)$ and since w is affine between f_i^* and f_{i+1}^* and h is convex between f_i^* and f_{i+1}^* , then $w \geq h$ on $[f_i^*, f_{i+1}^*]$.

We now show that $0 \leq \int_0^1 w(f) - h^*(f) d\mu \leq p/k$. First note that $\int_0^1 w(f) d\mu \geq \int_0^1 h^* d\mu$ because $w(f_i^*) = h(f_i^*) \geq g(f_i^*) = h^*(f_i^*)$ and both functions are piecewise affine between those points. For all i,

$$\begin{aligned} w(f_i^*) - h^*(f_i^*) &= h(f_i^*) - h^*(f_i^*) \\ &= h(f_i^*) - h(f_i) + h(f_i) - h^*(f_i^*) \\ &= h(f_i^*) - h(f_i) + g(f_i) - g(f_i^*) \\ &\leq |h(f_i^*) - h(f_i)| + |g(f_i) - g(f_i^*)| \\ &\leq 2p|f_i^* - f_i| \\ &\leq p/k. \end{aligned}$$

Furthermore, since w and h are piecewise affine, so is $w - h$. Its maximum absolute value between 0 and 1 is reached at one of the f_i^*. Therefore,

$$\begin{aligned} \int_0^1 w(f) - h^*(f) d\mu &\leq \int_0^1 \max_f (w(f) - h^*(f)) d\mu, \\ &\leq p/k \int_0^1 d\mu, \\ &= p/k. \end{aligned}$$

Finally, as soon as $p/k \leq \varepsilon$, we have

$$\int_0^1 w(f) d\mu \geq \int_0^1 h(f) d\mu \geq \int_0^1 h^*(f) d\mu \geq \int_0^1 w(f) d\mu - \varepsilon.$$

We further know that the solution $S_k^*(N, 0, 1)$ of the dynamic programming equation verifies $H(f_1^* \cdots f_N^*) \leq S_k^*(N, 0, 1) \leq H(f_1 \cdots f_N)$. This shows that $S_k^*(N, 0, 1) - H(f_1^* \cdots f_N^*) \leq \varepsilon$

Note that the bound on the value of k given by this proposition depends on g but does not depend on N so that k can be chosen uniformly for all values of N. Also note that since $[0, 1]^N$ is compact and $H(f_1, \cdots, f_N)$ is continuous in $f_1 \cdots, f_N$, the limits when k goes to infinity of all subsequences of the optimal solutions of the dynamic program are optimal solutions.

4 Several extensions

4.1 The uniform case

Here is one important particular case: the measure μ is uniform over $[0, 1]$. The uniform case corresponds to the case where one has no statistical information on the future use of the processor so that all speeds between 0 and 1 seem equally likely to be used.

In the uniform case, it is possible to give necessary conditions for $f_1 \cdots f_N$ to reach the minimum value of $S(N, 0, 1)$. Indeed, the cost function can be computed more explicitly (up to a multiplicative constant):

$$H(f_1 \cdots f_N) = \sum_{i=0}^{N} (f_{i+1} - f_i)(g(f_{i+1}) + g(f_i)).$$

By differentiating with respect to f_i, one gets (assuming that g is differentiable)

$$\frac{\partial H(f_1 \cdots f_N)}{\partial f_i} = -g(f_{i+1}) + g(f_{i-1}) + (f_{i+1} - f_{i-1})g'(f_i).$$

If $f_1 \cdots f_N$ is a local minimum of s, then for all $1 \leq i \leq N$,

$$g'(f_i) = \frac{g(f_{i+1}) - g(f_{i-1})}{f_{i+1} - f_{i-1}}. \tag{2}$$

If g is strictly convex and derivable, this means that f_i is the only point between f_{i-1} and f_{i+1} where the slope of g is the same as the global slope of g between f_{i-1} and f_{i+1} (see Figure 4).

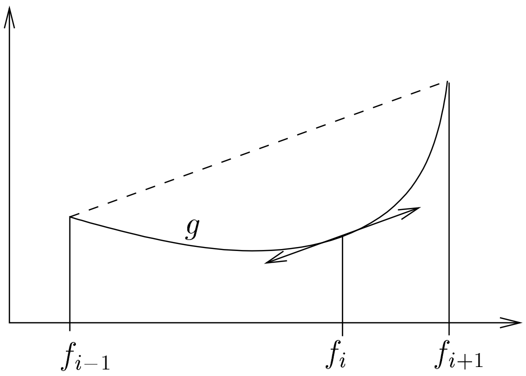

Fig. 4. The choice of speed f_i, once f_{i-1} and f_{i+1} are given

In some cases s has a unique local minimum (hence global) and the previous characterization can help to compute the minimum explicitly. However, in most cases s has many local minima and one needs additional properties to compute the global minimum of s. For instance, if g is a polynomial of degree d, then the system of equations for the local minima is made of n polynomial equations each of degree $d - 1$. The number of solutions can be as large as $(d - 1)^N$.

However, if we consider the typical case where the energy is quadratic [7], $g(x) = ax^2$ $(a > 0)$, then the system of equations (2) becomes:

$$
\begin{aligned}
2f_1 &= f_2 \\
2f_2 &= f_3 + f_1 \\
\vdots &= \vdots \\
2f_n &= f_{N+1} + 1
\end{aligned}
$$

Which has a unique solution $(\frac{1}{N+1}, \cdots, \frac{N}{N+1})$. This means that if the energy is quadratic in the frequency, then the speeds should be evenly distributed over $[0, 1]$ for any N.

4.2 The non convex case

There are two reasons to consider the case where g is not convex. The first one is by taking into account static power leaks of circuits which is more and more important nowadays and cannot be neglected. In this case the power consumption is not convex in the speeds any longer, as illustrated by figure 5.

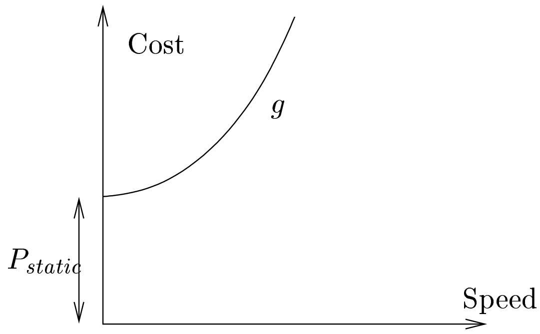

Fig. 5. Power consumption with a static leak P_{static}.

The second reason is because the cost function $g(f)$ should not only take into account the energy consumption using frequency f but also the hardware cost of implementing that frequency. In that case the cost g is also not convex in general.

When g is not convex, the circuit C introduced in Section 2.2 should not use the two neighboring speeds to emulate a given speed f in order to optimize its power consumption. Instead, its should use two frequencies f_i and f_j such that $f_i \leq f \leq f_j$ and all the points $(f_k, g(f_k))_{k=1\cdots N}$ lie above the straight line containing the points $(f_i, g(f_i))$ and $(f_j, g(f_j))$. This means that the average energy consumption rate is no longer the integral of the linear interpolation

$h(f_1 \cdots f_N)$ but it is integral of its lower convex hull, called $h_c(f_1 \cdots f_N)$ in the following (see Figure 6). Note that by definition of $h_c(f_1 \cdots f_N)$, $h_c(f_1 \cdots f_N) = h(f_{i_1}, \cdots, f_{i_K})$ where $\{f_{i_1}, \cdots, f_{i_K}\}$ is the subset of $\{f_1 \cdots f_N\}$ made of the extreme points used by the lower convex hull. For the example displayed in Figure 6, where $N = 3$, $h_c(f_1, f_2, f_3) = h(f_1, f_3)$ so that $K = 2$ and $i_1 = 1, i_2 = 3$.

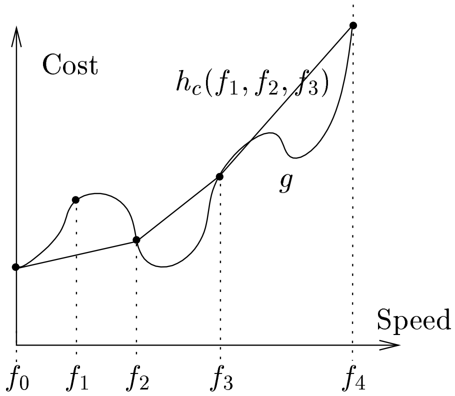

Fig. 6. The function h when g is not convex

The goal is now to choose the set of frequencies $f_1, \cdots, f_N$ for minimizing the integral $H_c(f_1, \cdots, f_N) = \int_0^1 h_c(f_1, \cdots, f_N) d\mu$.

As for the convex case, we discretize the problem by restricting the search to the finite set $\{n/k, n = 1, \cdots, k-1\}$.

We apply the same dynamic programming method as before. It should be clear that it provides a set of N frequencies $f_1^*, \cdots, f_N^*$ that minimize $H(f_1 \cdots f_N)$, as before. However, it is not clear that $H_c(f_1, \cdots, f_N)$ is also minimized by the same set $f_1^*, \cdots, f_N^*$. This is proved using the following lemma.

Lemma 1. *The lower convex hull of the points* $(f_1^*, g(f_1^*)), , \cdots, (f_N^*, g(f_N^*))$ *contains all the points* $(f_1^*, g(f_1^*)), \cdots, (f_N^*, g(f_N^*))$ *as extreme points (in other words,* $h(f_1^*, \cdots, f_N^*) = h_c(f_1^*, \cdots, f_N^*)$*).*

Proof. The proof is by contradiction. Let us suppose that the point $(f_a^*, g(f_a^*))$ is not an extreme point of the convex hull. Consider a new set of points, $f_1', \cdots, f_N'$ such that $f_i' = f_i^*$ for $i \neq a$ and $f_a' = f_{a+1}^*$. Then, $f_1', \cdots, f_N'$ is a valid solution to the dynamic programming problem and its cost is the integral of the linear interpolation of the points $f_1', \cdots, f_N'$ against μ: $\int_0^1 h(f_1', \cdots, f_N') d\mu$. This is obviously smaller than the cost of $f_1^*, \cdots, f_N^*$, namely $\int_0^1 h(f_1^*, \cdots, f_N^*) d\mu$ because by construction of $f_1', \cdots, f_N'$, $h(f_1', \cdots, f_N') \leq h(f_1^*, \cdots, f_N^*)$ everywhere.

Using the previous lemma, one may conclude that the dynamic program provides the best points for the minimization of the linear interpolation h which happens to be also convex and therefore coincides with h_c. This means that when using the dynamic program one gets the optimal solution for minimizing $H_c(f_1, \cdots, f_N)$. Indeed, for every $f_1, \cdots, f_N$, as mentioned above, there exists $K \leq N$ and $i_1, \cdots i_K$ such that

$$H_c(f_1, \cdots, f_N) = H(f_{i_1}, \cdots, f_{i_K}), \tag{3}$$

and

$$\begin{aligned} H_c(f_1^*, \cdots, f_N^*) &= H(f_1^*, \cdots, f_N^*), && (4)\\ &= \min_{f_1, \cdots, f_N} H(f_1, \cdots, f_N), && (5)\\ &= \min_{K=1 \cdots N} \min_{f_1, \cdots, f_K} H(f_1, \cdots, f_K), && (6)\\ &\leq \min_{f_1, \cdots, f_N} H_c(f_1, \cdots, f_N), && (7)\end{aligned}$$

where (4) holds because of Lemma 1, (5) is the definition of $(f_1^*, \cdots, f_N^*)$, (6) holds because one may choose $f_i = f_{i+1}$, and (7) is a consequence of (3).

Finally, this means that $f_1^*, \cdots, f_N^*$ also minimizes $H_c(f_1, \cdots, f_N)$.

As for computing a bound on k (the sampling parameter), Proposition 1 can also be applied here as long as g is continuous with left and right derivatives bounded by p everywhere.

4.3 Further extensions

Another factor that is very important in circuit design in order to reduce the energy consumption is to play on the hardware design to change the shape of the cost function g. We have too few informations at this point about the technological constraints that drive the construction of g to make a pertinent suggestion on this issue. The combined optimization of the shape of g and the choice of the N speeds is important in energy centric circuit design.

A practical implementation of this technique for real processors is required to state the gain with respect to easy heuristics.

5 Conclusion

In this paper, we present a dynamic programming technique to select the best N speeds for a power aware circuit with dynamically varying voltage. This computation is based on an *a priori* statistical knowledge of the likelihood of the frequencies.

References

1. F. Catthoor, S. Wuytack, E. De Greef, F. Balasa, L. Nachtergaele, and A. Vandecappelle. *Custom Memory Management Methodology.* Kluwer Academic Publishers, 1998.

2. A.P. Chandrakasan and R.W. Brodersen. Minimizing power consumption in digital cmos circuits. In *Proceedings of the IEEE*, volume 83, 1995.
3. B. Gaujal, N. Navet, and C. Walsh. A linear algorithm for real-time scheduling with optimal energy use. Technical Report RR-4886, INRIA, 2003.
4. F. Gruian. *Energy-Centric Scheduling for Real-Time Systems*. PhD thesis, Lund Institute of Technology, 2002.
5. I. Hong, M. Potkonjak, and M.B. Srivastava. On-line scheduling of hard real-time tasks on variable voltage processor. In *International Conference on Computer Design*, pages 653–656, 1998.
6. A. Qadi, S. Goddard, and S. Farritor. A dynamic voltage scaling algorithm for sporadic tasks. In *24th IEEE International Real-Time Systems Symposium*, pages 52–62, December 2003.
7. S. Saewong and R. Rajkumar. Practical voltage-scaling for fixed-priority rt-systems. In *9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2003.
8. Mohammed Es Salhiene, Laurent Fesquet, and Marc Renaudin. Dynamic voltage scheduling for real time asynchronous systems. In *Twelfth International Workshop on Power And Timing Modeling, Optimization and Simulation (PATMOS)*, pages 81–91, Sevilla, Spain, 2002.
9. F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of lEEE Annual Foundations of Computer Science*, pages 374–382, 1995.
10. Han-Saem Yun and Jihong Kim. On energy-optimal voltage scheduling for fixed-priority hard real-time systems. *ACM Transactions on Embedded Computing Systems*, 2(3):393–430, August 2003.