On the energy complexity of the FFT

†J. Smit and ‡J.A. Huisken

†University of Twente, Department of Electrical Engineering EF9250
POBox 217, NL 7500AE Enschede,e–mail: jaap@nt.el.utwente.nl
‡Philips Research Labs, bldg. WAY 41, Prof. Holstlaan 4,
NL 5656AA Eindhoven, e–mail: huisken@prl.philips.nl

Abstract

Methods for the calculation of a lower bound on the energy consumption of the realization of an algorithm in a CMOS will be introduced. A technology basis, consisting of the characterization of just three energy consumption figures, is used in the paper to determine the energy complexity of the FFT. A comprehensive set of concepts is introduced for the estimation of energy consumption levels of algorithms under constraints on placement, interconnect, rippling due to composition and energy consumption due to datatransport within memory systems. It will be shown, using these basic notions, that a partitioning of the FFT exists which dissipates an amount of energy due to arithmetic proportional to N log (N) and memory access proportional to $N\sqrt{N}$.a The amount of energy due to wiring is shown to be negligible w.r.t. the other terms. Moreover it is shown that the first term is relevant for values of $N < 2^{18}$ whereas the second term is relevant for larger FFTs.

Introduction

Low power design [1],[2] is a technique for the reduction of the energy consumption of a VLSI circuit which is operating at its full speed. *Power management* [3] tries to disable a temporarily unused part of say a CPU in order to adapt its energy consumption to the actual demand.

The introduction of CMOS VLSI circuits made low power design a seemingly obvious issue, as CMOS does not draw static current as opposed to many other logic families, like NMOS, TTL, ECL and the like. This is why the adjective low–power is still too frequently used without a specific meaning in product announcements and mostly algorithm specific publications.

Definitions and methods will be worked out in this article which make it possible to make a link between an *algorithm* and the total amount of energy needed to execute an energy optimal CMOS implementation of the algorithm. This statement implies that a method will be given to determine the *minimal amount of ener-*

gy needed to execute *a given algorithm*, including its arithmetic, layout and memory access implications.

Dissipation of energy in CMOS circuits

The three most important reasons for energy dissipation in a CMOS circuit are related to [2]:

1 Voltage transitions on capacitors.
2 The simultaneous conduction of p–and n– transistors.
3 Leakage of p–n junctions.

Establishing a technology reference

The amount of energy needed to transport a bit with a transition probability P_t over a wire with length l and characteristic capacitance $C_{/m}$ is: $\mathcal{E}w = P_t\ 1/2\ l\ C_{/m}\ Vw^2$. This formula equals: $\mathcal{E}w = P_t\ 1/2\ l\ C_{/m}\ V_{DD}^2$ when the voltage on the wire Vw makes a full swing to V_{DD}.

	Value	Unit
$\mathcal{E}_w$	$P_t\ 1/2\ l\ C_{/m}(\lambda)\ V_{DD}^2$	[J]
$\mathcal{E}_{fa}$	$\mathcal{E}_{fa}(\lambda, V_{DD})$	[J]
$\mathcal{D}_{cell}$	$\mathcal{D}_{cell}(\lambda)$	[m]

Table 1.

The lowest average amount of energy needed to add three bit–values, into a sum and carry is: $\mathcal{E}_{fa}$. The actual value of $\mathcal{E}_{fa}$ depends on many factors, including the finest geometrical detail λ, the supply voltage V_{DD}, the proper tapering of all transistors in the design and the use of safe operating margins within the cell design. We have developed an automatic characterization program for an arbitrary full–adder design. This makes it possible to optimize CMOS circuits at the technological basis on one hand, while stressing the independence of technology of any predicted lower bounds involved in the energy complexity of algorithms like the FFT. The third parameter in the technology reference is the diameter of a cell in a random access memory. This value is of great importance for the prediction of the energy consumption related to datastorage.

Calibration of the reference

The amount of energy dissipated due to datatransport over a wire of l [m] is: $\mathcal{E}w = l\ P_t\ 1/2\ C_{/m}\ V_{DD}^2$. This amounts to $\mathcal{E}w = l\ P_t\ 1.44$ nJ, with Pt the transition probability of the signal, for a given 1μm 5V CMOS process.

A schematic diagram of a well known full–adder design is presented in figure 1. Figure 2. gives an optimally tapered layout for the given full–adder schematic with

respect to minimal energy consumption, good timing and robust electrical behavior under normal production tolerances. It is not excluded that marginally better designs can be presented, however the qualification of a design to all of the parameters mentioned goes beyond the scope of this paper.

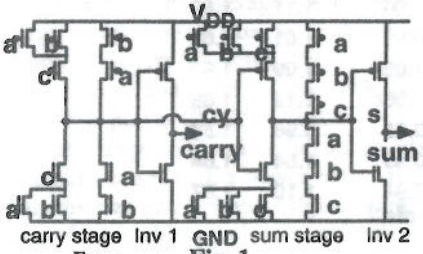

Fig. 1.

To a b c / From a b c	000	001	010	011	100	101	110	111
000		3.07	3.18	2.70	3.23	2.73	2.84	5.39
001	2,31		0.51	6.63	0.58	6.63	6.93	7.29
010	2.16	0.00		6.25	0.30	6.32	6.43	6.41
011	4.11	4.41	4.57		4.69	0.27	0.51	2.85
100	2.18	0.00	0.07	6.15		6.23	6.31	5.67
101	4.41	4.26	4.36	0.09	4.44		0.33	2.55
110	4.73	4.29	4.34	0.07	4.48	0.16		2.45
111	3.88	1.57	1.70	2.26	1.83	2.33	2.49	

Table 1.

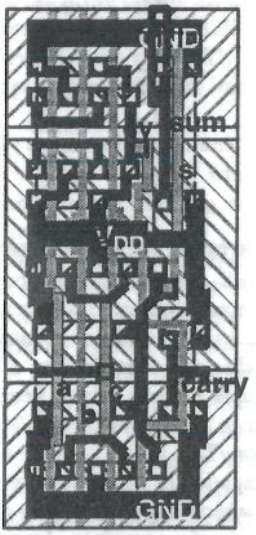

Fig. 2.

I.e. the 1μm 5V CMOS design presented in figure 1 is considered to be a good one, but any design team which can improve on it can derive and characterize it using SPICE simulations ór the like, in the form of a table for a full–adder, loaded on the Cout output with a Cin–input and on the Sum output with the b–input, which monitors the energy consumption as a function of the state transitions on the input signals: a, b and c, like the one shown in table 1.

Table 1 contains the energy consumption in pJ for all possible state transitions of the full–adder realized in a 1 μm CMOS process.

The average energy consumption g_{fa}

The actual energy consumption of the full adder presented in table 1 is by far too complex to be easily handled in an energy complexity analysis of a design. The perfect symmetry of the full adder makes it however simple to calculate the aver-

age energy consumption $\mathcal{E}_{fa}$, as shown in table 2, together with the average delay τ and the maximum delay t for the Sum and the Cout signal.

	Energy [pJ]	Timing cout [ns]		Timing sum [ns]	
		τ (avg)	t (max)	τ (avg)	t (max)
$\mathcal{E}_{fa}$	2.41	0.67	1.06	1.11	1.69
$\mathcal{E}_{a}$	1.92	0.43	0.99	1.01	1.65
$\mathcal{E}_{b}$	2.25	0.62	1.05	1.09	1.68
$\mathcal{E}_{c}$	2.67	0.78	1.06	1.12	1.68
$\mathcal{E}_{ab}$	1.68	0.38	0.88	0.96	1.54
$\mathcal{E}_{bc}$	2.27	0.56	0.93	1.04	1.69
$\mathcal{E}_{ac}$	2.54	0.72	0.94	1.10	1.57

Table 2.

The average energy consumption $\mathcal{E}_{xy}$

The full–adder is used in some applications in such a way that either one or even two inputs, indicated with a subscript, are kept constant for a prolonged period. Table 2 shows additional entries for the average energy consumption of the full adder when any combination of inputs is kept constant.

Example circuits which keep one or two inputs constant for some time are for instance the ripple–carry adder, which initially performs a normal full–adder operation. This initial full adder operation is subsequently followed by a carry–ripple propagation in which the a–input as well as the b–input are kept constant. A formula in which $\mathcal{E}_{ab}$ is explicitly referenced is considered too detailed for a complexity analysis of a complete algorithm. Hence we introduce a ripple factor ϱ which reflects, as a function of the architecture of an high level building block at hand, the factor which reflects the extra energy consumed due to rippling in the datapath.

Energy consumption due to wiring and datastorage

The total energy complexity is calculated as the sum of the arithmetic energy complexity $\mathcal{E}_{ar}$, the energy complexity due to wiring $\mathcal{E}_{w}$ and the energy complexity due to data–storage $\mathcal{E}_{RAM}$.

Total energy complexity: $\mathcal{E}_{tot} = \mathcal{E}_{ar} + \mathcal{E}_{w} + \mathcal{E}_{RAM}$

Power radius: $\mathcal{R}_{pd} = \mathcal{E}_{cell} / \mathcal{E}_{w/m}$

Diameter of a RAM–cell: $\mathcal{D}_{cell} = \mathcal{H}_{cell} + \mathcal{W}_{cell}$

Diameter of a RAM: $\mathcal{D}_{RAM} = \sqrt{N}\, \mathcal{D}_{cell}$

Ideal energy consumption of a RAM: $\mathcal{E}_{ideal} = \mathcal{D}_{RAM}\, \mathcal{W}\, P_t\, \mathcal{E}_{/m}$

Overhead efficiency of a RAM: $\eta_{ov} = \mathcal{W} / (\mathcal{W} + \mathcal{A})$

Access efficiency of a RAM: $\eta_{acc} = \mathcal{E}_{ideal} / (\eta_{ov}\, \mathcal{E}_{RAM})$

Actual energy consumption of a RAM: $\mathcal{E}_{RAM} = \mathcal{E}_{ideal} / (\eta_{ov}\, \eta_{acc})$

The power radius of a cell $\mathcal{R}_{pd}$ defines the length of the wires connected to its terminals such that the average energy consumption due to wiring equals the average energy consumption due to arithmetic. The power radius of the full adder introduced is just 0.67 mm. This is 10 x its height and 30 x its width. The rather small absolute and relative value of $\mathcal{R}_{pd}$ indicate that placement plays an important role in low power CMOS circuits.

Designs with primitives like full adders placed closely together in arithmetic building blocks can realize a power radius at the next level in the hierarchy which exceeds the size of the whole chip. This makes it possible to simplify: $\mathcal{E}_{tot} \approx \mathcal{E}_{ar} + \mathcal{E}_{RAM}$, for an arbitrary placement for the given building block.

The efficiency η_{ov}, with which a Random Access Memory can be used depends on the number of address bits $\mathcal{A}$ which is used in conjunction with the number $\mathcal{W}$ of data bits, as far as the overhead due to addressing is concerned, and the access efficiency η_{acc}, as far as read/write operation to the memory system are concerned. The definitions presented are simple and effective. A good RAM should have an access efficiency η_{acc} which comes close to 1. The architecture of the 6–transistor SRAM cell, with its precharge & selective discharge protocol makes it however hard to obtain values for $\eta_{acc} > 1/8$. Hierarchical RAM structures may however be designed which realize η_{acc} values which come close to 1, although speed and compactness considerations tend to give lower values. Application of the formulas to a 512k x 8 low–power 23–ns SRAM [9], an SRAM with hierarchical sense amplifiers [10] and a 2M x 8 12–ns SRAM, gives η_{acc} values of 0.091, 0.024 and 0.026 respectively.

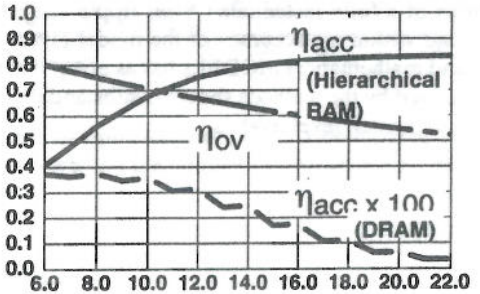

Fig. 3.

Figure 3. shows the value of η_{ov} and η_{acc} for realistic RAMs of various construction, with a size ranging from 2^6 24–bit words to 2^{22} 24–bit words. Note that it is sufficient for the calculation of the complexity of an algorithm to take a RAM with an access efficiency which comes close to 1.

Composition related energy dissipation

Structural composition has side effects on the amount of energy dissipated due to the introduction of glitching in the circuit (cells & interconnect). The carry–ripple adder of figure 4. is an instance of a circuit in which glitching plays a role which cannot be neglected. We use the energy consumption tables of the full–adder to calculate the switching energy accurately in the average case, as opposed to the method based on the ratio of useful and useless transitions introduced in [4].

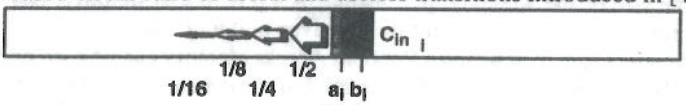

Fig. 4.

The energy consumption originating from a single bit of a carry–ripple adder, located k–bits from the msb of the adder, $\mathcal{E}_{cr}$ is:

$$\mathcal{E}_{cr}(k) = \mathcal{E}_{fa} \qquad k = 1$$

$$\mathcal{E}_{cr}(k) = \mathcal{E}_c + \sum_{m=0}^{k-1} 1/2^m\, \mathcal{E}_{ab} = \mathcal{E}_c + (1 - 1/2^k)\mathcal{E}_{ab} \qquad k > 1$$

Figure 5. shows this value for the reference full–adder as a function of m.
The energy dissipated by an m–bits wide carry–ripple adder is:

$$\mathcal{E}_{ripple}(m) = \sum_{k=1}^{m} \mathcal{E}_{cr(k)} = m\, Q_{ripple}\, \mathcal{E}_{fa}$$

SIPCE simulations of a fully instantiated 4–bit ripple carry adder reveal that the predicted results are within the accuracy of the model given. The combination of accurate SPICE and mathematical MODELs has as effect that the energy overhead due to rippling, derived by the methods shown is substantially lower than predicted in [4] on the basis of switch level models.

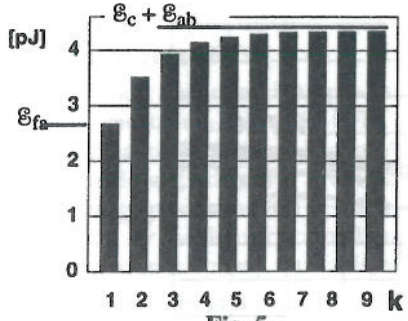

Fig. 5.

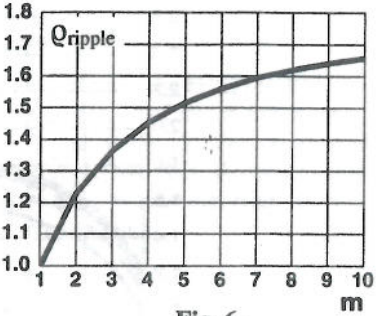

Fig. 6.

Extension of the method presented for 1D ripple adders in the figures 5. and 6., to the 2D ripple–adder structures, of figure 7. gives a ripple factor $Q_{cascade}$ as a function of the number of bits m, per word and the number of additions n as shown in figure 8.

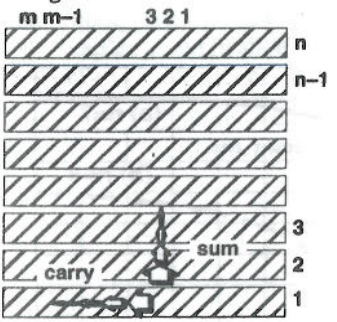

Fig. 7.

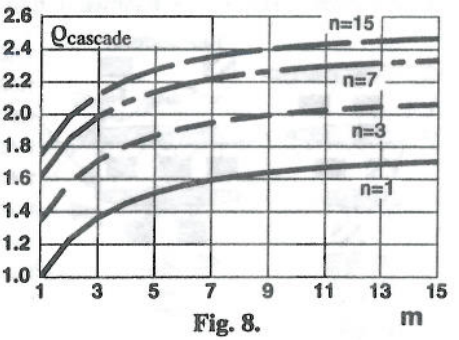

Fig. 8.

The use of a binary tree of ripple–adders gives a ripple factor as a function of the number of bits m, per word and the number of additions n as shown in figure 9.

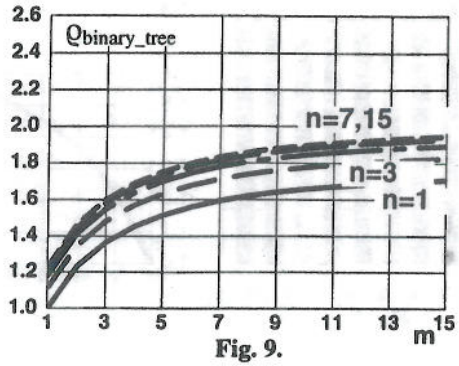

Fig. 9.

The implementation of a circuit to reduce glitches in a Wallace tree multiplier introduced in [4] and shown in figure 10.a, has a ripple factor shown in figure 11.a, whereas a hybrid approach with a reduced logic depth and a final cascade adder has the improved ripple factor shown in figure 11.

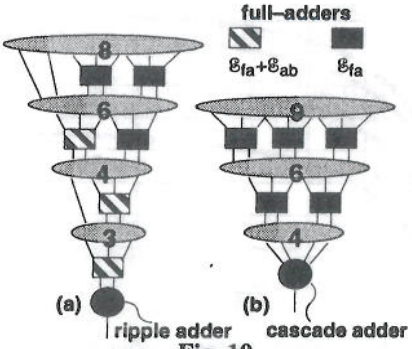

Fig. 10.

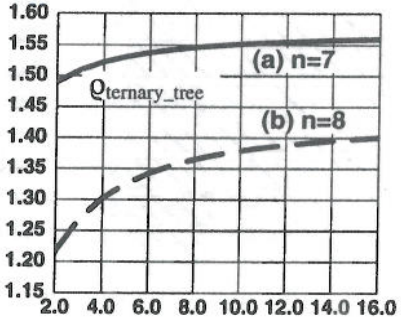

Fig. 11.

A more detailed analysis of the effect of composition on the overall overhead in various combinatorial circuits can be found in [11].

The energy consumption of an n x m multiplier can be writen as:

$\varepsilon_{mul} = n\ m\ (Q_{mul}\ \varepsilon_{fa} + \varepsilon_{and})$

The ripple factor Q_{mul} depends weakly on the size of the multiplier and has a value

which lies, depending on the architecture, between 1.37 and 2.25 for typical 8x8 multipliers.

Applications of the FFT

One of the most important signal processing algorithms is the FFT [13]. Its low–energy implementation, called the EFFT in this article, is of major importance for many application areas, like: Digital Audio Broadcasting [5], Digital Terrestrial Television Broadcasting [7], Synthetic Aperture Radar [12], Reconstruction in medical image processing (CT & MRI), Audio signal processing, High speed echo cancellation for the delivery of multimedia over existing telephone wiring, Conversion between telephony standards, etc.

Energy complexity of the classical FFT

The basic algorithm executed by the FFT is [13],[14]:

$$X[k] = \sum_{n=0}^{N-1} x[n] \; e^{-j(2\pi/N)kn} = \sum_{n=0}^{N-1} x[n] \; W_N^{kn}$$

The arithmetic building block shown in figure 12. is the so called *Butterfly*. It contains a complex multiplication one complex addition and one complex subtraction. The complex multiplication is typically calculated as:

$(a + j\,b) \times (c + j\,d) = (a\,c - b\,d) + j\,(a\,d + b\,c)$

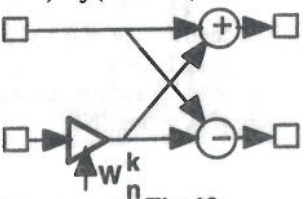

Fig. 12.

The amount of energy consumed by the classical signed complex multiplier is four times, the amount consumed by a classical signed real multiplier, plus the amount of two double precision additions. The butterfly datapath can be realized, given the large $\mathcal{R}_{pd}$ of the multiplier, with a total energy complexity:

$\mathcal{E}_{butterfly} = \mathcal{E}_{tot} \approx \mathcal{E}_{ar}$:

$\mathcal{E}_{butterfly} = 4\,m\,[n\,(Q_{mul}\,\mathcal{E}_{fa} + \mathcal{E}_{and}) + Q_{mul}\,\mathcal{E}_{fa}]$

where m is the width of the main datapath and n the width of the twiddle factors.

Each of the N/2 $^2\log(N)$ butterflies has 2 inputs and 2 outputs which are usually read from, and stored in, a RAM.

$\mathcal{E}_{FFT} = \mathcal{E}_{butterfly} + \mathcal{E}_{RAM}$
This gives:
$\mathcal{E}_{FFT} \approx 2N\ ^2\log(N)[n\ m\ (Q_{mul}\ \mathcal{E}_{fa} + \mathcal{E}_{and}) + \sqrt{N}\ \mathcal{D}_{cell}\ \mathcal{W}\ P_t\ \mathcal{E}_{/m} / (\eta_{ov}\ \eta_{acc})]$
This gives for $\mathcal{W} = 2$ m, $\mathcal{A} = {}^2\log(N)$, $P_t = 1/2$ and the realizable value: $\eta_{acc} = 1/2$:
$\mathcal{E}_{FFT} / N \approx 2\ ^2\log(N)\ [n\ m\ (Q_{mul}\ \mathcal{E}_{fa} + \mathcal{E}_{and}) + \sqrt{N}\ \mathcal{D}_{cell}\ (2m + {}^2\log(N))\ \mathcal{E}_{/m}]$

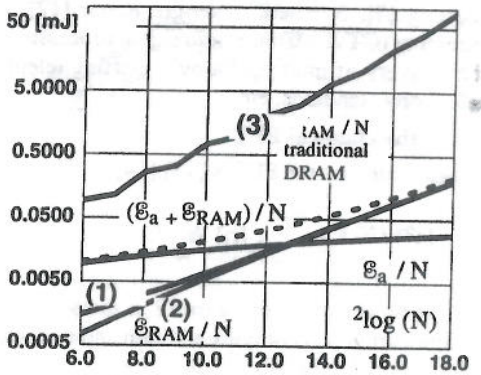

Fig. 13.

Figure 13. shows the normalized contribution of RAM access and arithmetic to the energy complexity of the classical FFT for a 1μm process with n=8, m=12 for a well designed RAM (1) an ideal RAM (2) and a traditional DRAM (3) from a given design library for various values of $\mathcal{A} = {}^2\log(N)$.

The EFFT

The contribution of RAM access to the energy complexity of the FFT is so substantial for $^2\log(N) > 10$ that a new algorithm which consumes less energy due to memory access becomes a must for low power applications. The number of memory cycles can be reduced by implementing a larger radix, like for instance 4 or 8, or by rearrangement of the operations within the FFT, such that a much smaller RAM may be used for the intermediate operations.
The permutations within the FFT are mapped onto wiring in the first option, whereas they are mapped on a memory system in the second option. The fact that a memory cell is much smaller than a complex multiplier (or a butterfly), indicates that the second option is the preferred method to reduce the energy consumtion of the FFT.

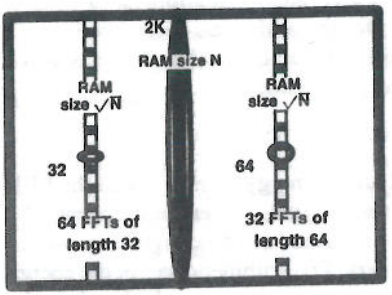

Fig. 14.

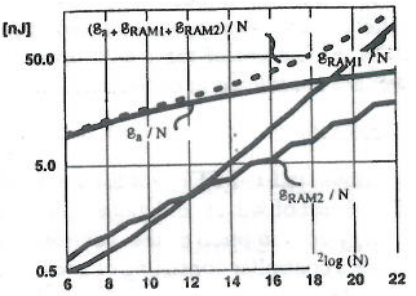

Fig. 15.

Figure 14. is a symbolic representation of the partitioning of a realization of the EFFT of length 2K. The EFFT executes in this case 64 FFT's of length 32 and 32 FFTs of length 64. The small FFTs use RAM of length 32 and 64 respectively, in most cycles. Just the first and last stage of all small FFTs access a RAMs of size N. The total amount of cycles to a RAM of size N is hence 4N, whereas the number of cycles to a RAM of size $\mathcal{O}(\sqrt{N})$ is: $4N\,(^2\log(N) - 1)$ for $N \geq 4$.

Figure 15. shows all contributions to the energy consumption of the EFFT for same parameters as used in figure 13. Note that the energy consumption due to the small RAMs: $\mathcal{E}_{RAM2} / N$, is substantial for small values of $^2\log(N)$ and negligible for $^2\log(N) \geq 12$.

It is possible to apply the mapping of the FFT in the innerloop on an even smaller RAM, however the results shown in figure 15. indicate however that such a mapping is attractive only for rather small values of $^2\log(N)$. The energy complexity of the EFFT is hence:

$$\mathcal{E}_{EFFT} / N \approx 2\,^2\log(N)\,[n\,m\,(\mathcal{Q}_{mul}\,\mathcal{E}_{fa} + \mathcal{E}_{and})] + 4\sqrt{N}\,\mathcal{D}_{cell}\,(2m + {}^2\log(N))\,\mathcal{E}_{/m}]$$

Applications

The channel decoder of a DAB receiver [5] contains an FFT operating at 12MHz. The lower bound of the power consumption for a 2K EFFT, suitable for Digital Audio Broadcast (DAB) applications, running at a sample rate of 1MHz, realized in the 1μm 5V CMOS process given, with n=8 and m=12 is: $22\ 10^{-9}$ [J] x $1\ 10^{6}$ [Hz] = $22\ 10^{-3}$ [W]

An EFFT suited for Digital Terestial Television Broadcasting (dTTb) applications [6] runs with the same parameters at a sample rate of ~16MHz. It has a lower bound on the power consumption of: $22\ 10^{-9}$ [J] x $16\ 10^{6}$ [Hz] = $352\ 10^{-3}$ [W]. This latter implementation is feasible in the technology given using radix–8 hardware

to keep the memory speed sufficiently low. Two complex multipliers, running in parallel are used to support the speed needed for the arithmetic operations.
CMOS realizations of these algorithms which consume less than twice as much power are considered to be feasible under the given conditions.

Remarks

- A simple radix–2 FFT was used to establish the energy complexity of the FFT. The effect of wiring and memory access was explicitly taken into account. The energy consumption of the controller was assumed to be negligible, as we assume a controller consisting of a few counters. The arithmetic part of the energy complexity plays a major role for $^2\log(N) < 18$. Techniques which reduce the arithmetic energy complexity without introducing too much control overhead may still be considered to improve the results given.
- It is known that a complex multiplication can be performed with just three multiplications instead of the four used for the calculation of the arithmetic complexity. Such realizations require however much more adders/subtractors. These extra adders/subtractors make it non–trivial to come with a realization which consumes neglible wiring energy.
- Asynchronous circuits realize a given function together with an isochronous region, similar in a way to combinatorial circuits together with registers. Asynchronous realizations of any form of the EFFT were not considered so far.
- The theory presented extends to asynchronous circuits as well when it can be (dis–)proven that there exists an asynchronous realizatiuon of a (complex) multiplier which draws less power than its combinatorial counterpart. The energy consumption of an asynchronous multiplier can be expressed in terms of the energy consumption of the full adder using: $\mathcal{E}_{asyn\_mul} = n\,m\,(Q_{asyn}\,\mathcal{E}_{fa} + Q_{and}\,\mathcal{E}_{and})$ The factors Q_{asyn} and Q_{and} 'compensate' for the extra energy needed to establish isochronity at the arithmetic level. Both techniques can be compared on the basis of their respective ripple factors, as the asynchronous circuit will not introduce ripples at the arithmetic level. An asynchronous realization is hence preferable for large m and n, iff: $Q_{asyn} < Q_{mul}$
- Similar remarks can be made for multiplier realizations, not explicitly studied in this paper, like for instance the booth multiplier, etc.
- Special purpose low power full–adder cells were used in a given process using a dedicated multiplier layout. The use of STD–cells for the construction of multipliers, using a well tuned library gives an increase in the arithmetic energy consumption with a typical factor of 2 to 3.

- It is mandatory to use high quality Random Access Memory structures designed for low power applications, like SRAMs, register banks, DRAMs etc., to get a chance to realize the EFFT closely to its energy complexity limit.

Conclusions

A theory and methodology for the estimation of the energy complexity of an algorithm realized in CMOS was introduced.
Strong statements about the minimal amount of energy needed to execute a realization of a given algorithm in CMOS could be made using these concepts. A new, power optimal, variant of the FFT, the EFFT could be identified with an energy complexity of:

$$\mathcal{E}_{EFFT} = \alpha N^2 \log(N) + \beta N \sqrt{N}$$

Acknowledgements

The authors want to thank Harry Binnendijk and Wouter Bronswijk for their work concerning the characterization of the full–adder using SPICE simulations and Bertil Janssen for the verification of the glitching behaviour of various full–adder configurations for a wide range of stimuli.

[1] R. Brodersen, A Chandrakasan and S. Sheng, "Low–Power Signal Processing Systems", 5th IEEE Workshop on VLSI Signal Procvessing, California, USA, Oktober 1992.

[2] N.Weste and K. Esraghian, Principles of CMOS VLSI design. ISBN 0–201–08222–5.

[3] B. W. Seussmith and G. Paap III, "Power PC 603 Power Management", CACM, June 1994, Vol 27, No. 6, pp. 43–46.

[4] J. Leijten, J van Meerbergen and J. Jess, "Analysis and Reduction of Glitches in Synchronous Networks", Proc. EDAC 1995.

[5] J.A. Huisken, F v.d. Laar, A Delaruelle and N.J.L. Philips, *Specification, Partitioning and Design of a DAB channel decoder*, VLSI signal processing VI, ISBN 0–7803–0996
–0, pp 21–29.

[6] P.G.M de Bot et. al. "An overview of the modulation and Channel Coding Schemes Developed for Terrestrial Television Broadcasting within the dTTb project.", private communication.

[7] E.Bidet, C. Joanblanq, P. Senn. *A Fast Single Chip Implementation of a 8192 Complex points FFT.* IEEE 1994 Custom integrated Circuits Conference, pp 207–210.

[8] T.K. Callaway and E.E. Schartzlander, Jr, "Optimizing Arithmetic Elements for Signal Processing", 5th IEEE Workshop on VLSI Signal Procvessing, California, USA, Oktober 1992.

[9] K. Kimura, K. Itoh, R. Hori, J. Etoh, Y Kawajiri, H. Kawamoto, K. Sato and T. Matsumoto, "Power Reduction Techniques in Megabit DRAM's", IEEE J. of Solid State Circuits, 21, #3 Jun 1986

[10] K Sasaki, K. Ishibashi, K. Shimohigashi, T. Yamanaka, N. Moriwaki, S. Honjo, S. Ikeda, A. Koike, S. Meguro and O. Minato. "A 23–ns 4–Mb CMOS SRAM with 0.2–μA StandbyCurrent. IEEE J. of Solid State Circuits, 25, #5 Oct 1990.

[11] J. Smit, Internal report 076N95, University of Twente.

[12] J.Dall, J.Jorgensen, and S.N.Madsen, *Real–time processor for the Danish airborne SAR*, IEE Proc–F 139(2), pp115–121, April 1992.

[13] A.V. Oppeheim and R.W.Schafer, *Digital Signal Processing, Prentice Hall, 1975,* Prentice–Hall International Editions, ISBN 0–13–214107–8 01.

[14] L.R. Rabiner and B Gold, Theory and Applications of Digital Signal Processing, Prentice Hall, pp600–614, 1975.