Low-Power Analysis Using ORINOCO®[1]

By Dr. Stanley J. Krolikoski, Dr. Wolfgang Nebel and Dr. Laila Kabous

1. Introduction

Designing for lower power has become the hot topic of the day not only at electronics conferences and tradeshows, but down in the trenches where designers are being told to cram more functionality into smaller designs that must run faster than last year's designs. And, oh yes, these new designs, especially in the consumer electronics market, must when implemented have low power usage profiles to minimize battery usage.

This article discusses a system-level methodology offered by ChipVision Design Systems that seeks to identify potential "power bottlenecks" as early in the design process as possible, specifically at the "Electronic System Level" ("ESL"). There is no claim that lower level power analysis and remediation will no longer be required. Rather, the notion is that lower level power analysis and optimization tools can be more efficiently be used if a large number of power-related issues are handled at higher levels of abstraction.

2. Where ORINOCO fits in the design flow

ORINOCO is used as part of a HW design methodology in which algorithms are initially captured in C or SystemC and eventually are implemented (often as blocks in a larger design) using a logic synthesis-based flow. ORINOCO operates upon the C/SystemC® algorithms before implementation in an effort to reduce as many power usage issues as possible before implementation begins.

ORINOCO works optimally on designs that are data-dominated, as are typically found in signal processing applications. This ought to be not surprising, given that devices that do lots of data processing (often signal processing) such as wireless and multimedia devices, have severe power-related issues that result most notably in limited time between battery recharges or replacement. Indeed, ORINOCO fits into this part of the design spectrum quite nicely, since designers in the wireless and multimedia world have long used an ESL methodology using tools such as MATLAB®, SPW® and Cossap®.

3. Power Analysis Without Orinoco

The traditional approach to designing for lower power is to estimate and analyze power consumption in designs at the register transfer level (RTL) or the gate level, and to modify the design accordingly. In the best case, only the RTL within given functional blocks is modified, and the blocks re-synthesized. The process is repeated until the desired power results are achieved. This is shown in figure 1.

Unfortunately, the desired power consumption reductions may often be achieved only by modifying the basic architecture – and even the algorithms – of the design. However, modifications at this level affect not only power consumption, but also other performance metrics, and may actually significantly affect the economics of the chip. Thus, such modifications require re-evaluation and re-verification of the entire design, followed by a re-synthesis of the design.

[1] ORINOCO is a registered trademark of ChipVision Design Systems. All other trademarks and registered trademarks mentioned in this paper are the intellectual property of their respective owners

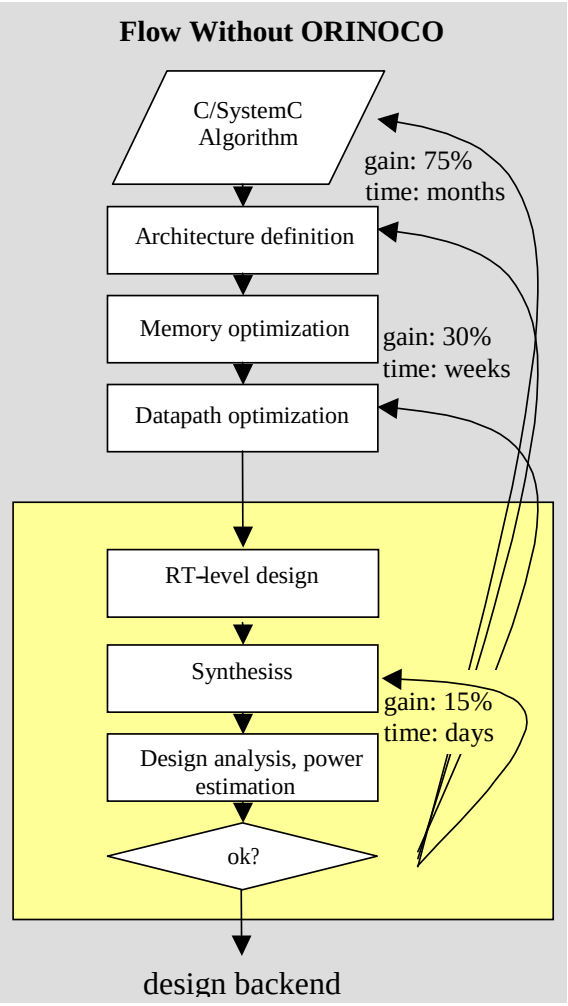

Figure 1: Power Analysis Without ORINOCO

Relatively simple modifications upon existing designs, such as datapath optimization, may achieve power reductions of the order of 30%, with a delay of several weeks. However, a more far-reaching architectural and algorithmic re-design necessary to reduce power by up to 75% may take months.

There is another less obvious issue in the case where reiteration to the higher level is necessary—availability of the right people to do the necessary algorithmic/architecture changes. Very few companies have a methodology where a system-level design team sits waiting to see whether the implementation team has issues for them. Indeed, it is usually the case that team that designed the front end will mostly have gone onto a new project once they handed the design off to the implementation team. Reassembling front end team to redo the data path or general architecture and algorithmic rework may, thus, be nearly impossible.

4. ORINOCO- The ESL Power Analysis Solution

Figure 2 shows an alternative to the situation where design iteration, including potential "reverse hand off" back to the front end team. Simply put, this flow is based around the notion that the earlier that power tradeoffs can be made, the better. Such early power analysis promises benefits both in terms of the speed at which these tradeoffs can be made, and the avoidance of costly front-end/back-end reiterations.

The system specification defines system requirements, and is expressed at a very high level of abstraction. This specification is usually written in a standard language, such as C/C++ or SystemC.

Using this system specification, algorithms that realize system functionality are developed and optimized, generally in those same standard languages. The algorithmic description consists of an executable specification, or functional description. It can be written as a behavioural description, which can be refined into a bit-accurate, pure functional design description.

After this the architecture, i.e., memory, controller and the data path structure, is developed to implement the given algorithms. Of course, while during such architectural development multiple design constraints, such as power, as well as performance and area ought to be taken into account—but, of course, often are not at the ESL—often because there are no structured tools available to carry out such analysis.

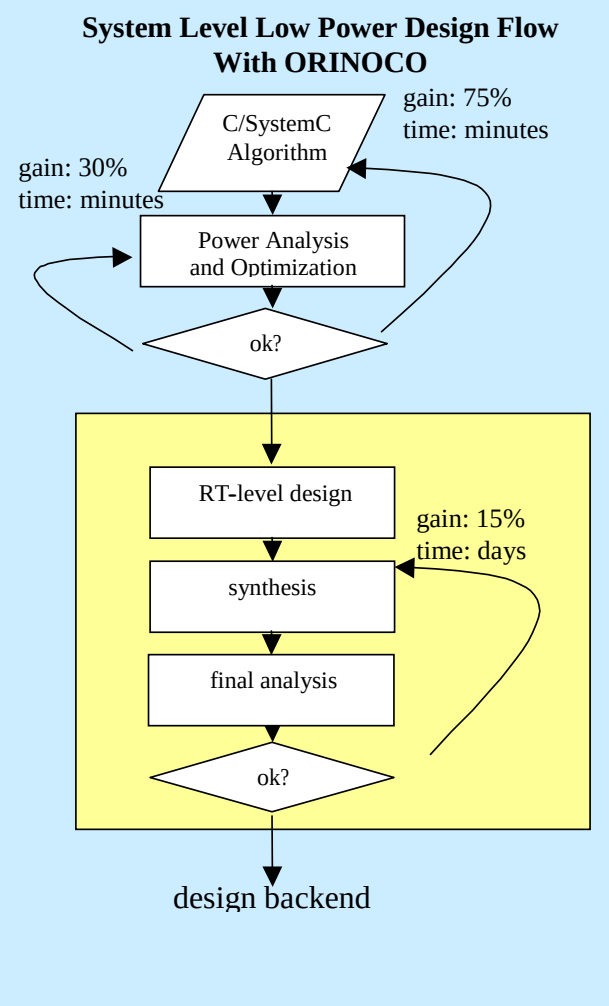

Figure 2: System-Level Low Power Design Flow

In ESL design using ORINOCO, power-optimal algorithms and architectures are developed according to the methodology flow in figure 3. It should be noted that system-level optimization targets dynamic power consumption, P_{load}, which is calculated from four factors, namely, clock frequency, the square of the supply voltage, load capacitance and average switching activity. Short-circuit power and leakage powers are optimized at lower levels of abstraction.

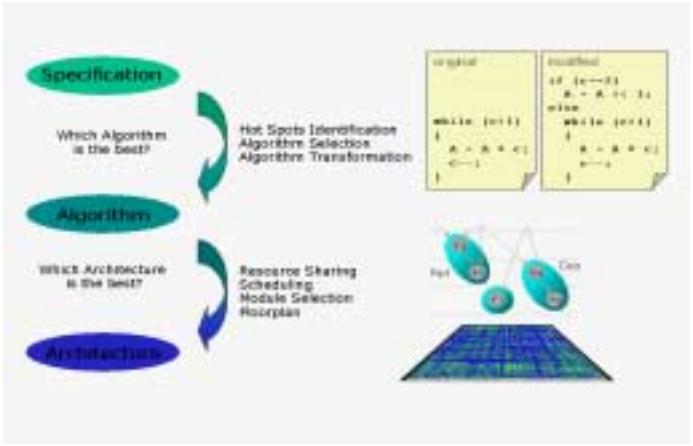

Figure 3: Design of Power-Optimal Algorithms and Architectures

Candidate algorithms are analyzed for their power characteristics, and to identify potential function-level hot spots. The most promising algorithms are then selected and optimized. This is then followed by creation of a power optimal system architecture. The optimal power-consuming functions are then transformed into hardware. The process is an iterative one of power estimation and optimization, with each iteration consuming minutes or hours, rather than days or weeks.

4.1 Algorithm Analysis and Optimization

In ORINOCO, the best algorithm to implement the specification is selected from the most appropriate candidate algorithms. “Power Bottlenecks” of each algorithm are identified and the algorithm optimized by algorithm transformation, as appropriate.

Analysis of the power consumption for a given algorithmic specification proceeds according to the flow shown in figure 4. The C/SystemC specification must first undergo compilation and instrumentation. Instrumentation is the process of inserting the profiling statements necessary to derive the switching activity at each defined operation in the source code. The algorithms are then executed, and the resulting activity profile data is used to annotate a suitable

design representation, a control data flow (CDF) graph. Any number of power estimations may then be performed to determine the power characteristics of any given configuration.

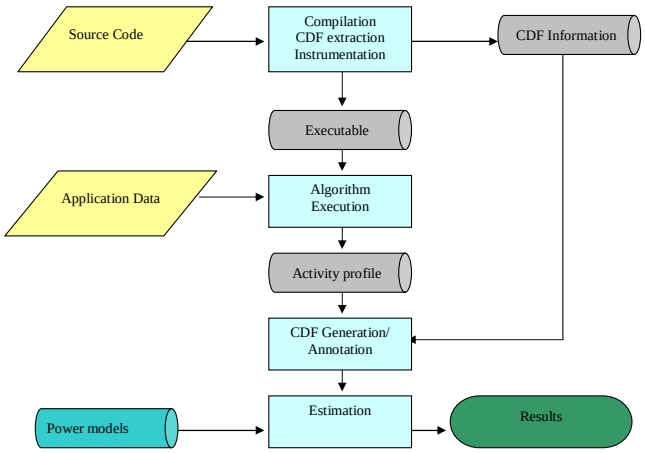

Figure 4: Algorithm Power Estimation Using a Control Data Flow Graph

A power-optimized architecture can be derived in ORINOCO from this graph without executing a complete synthesis, utilizing power models created for each RT-level component. These models depend on the input data, component characteristics such as bit width and architecture, and the underlying technology or cell library. The power models may be generated automatically for a given technology. Using the switching activity and the power models, the power consumption of a component can be estimated.

Algorithm transformation techniques in ORINOCO include operator substitutions, and code transformations. Code transformations include transformations on conditional statements, loop splitting, and loop unrolling. Control statement reduction has a significant impact on the power consumption, and such transformations are often best effected via loop transformations. An example of the power reduction effects of loop unrolling and common case optimization techniques is shown in figure 5. The results show the power consumption for the original algorithm (matrix_simple), the algorithm optimized with loop unrolling (matrix_unrolled), and the matrix optimized by common case techniques, (matrix_ccase_opt).

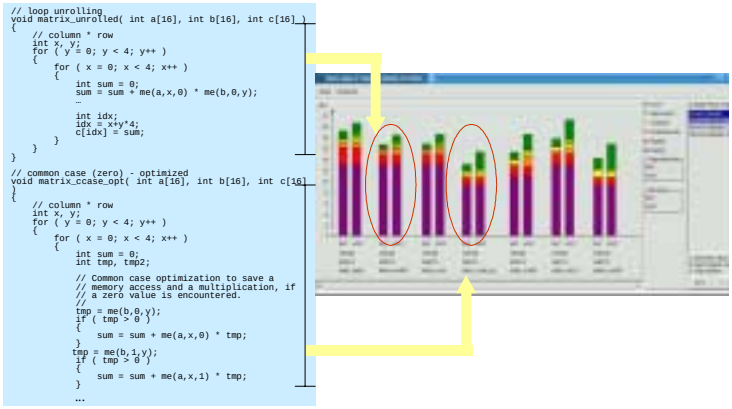

Figure 5: ORINOCO Power Analysis Before and After Loop Unrolling

An example of how the best algorithm can be selected in ORINOCO using the initial specification and the input data stream is shown in figure 6. The benchmark consists of two JPEG decompression algorithms, one in which very little data is lost during compression, and one that is faster but more lossy. The benchmark compares the energy consumption of the two algorithms resulting from the processing of two different input streams: a high quality stream – with a low compression ratio – consisting of 99% of the original data, and a fast stream consisting of 30% of the original data.

It can be seen that the fast (lossy) algorithm use less power consumption, while the accurate algorithm consumes more power when processing the fast stream. Analysis of such algorithms without ORINOCO would have been messy—to say the least, and would have taken weeks—as opposed to minutes with ORINOCO.

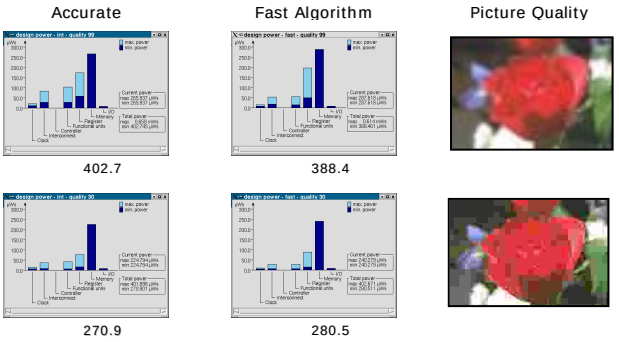

Figure 6: Energy Consumption of Two JPEG Algorithms in ORINOCO

4.2 Architecture Analysis and Optimization

Algorithmic estimation and optimization are followed in ORINOCO by the creation of a power-optimal architecture. This includes memory architecture, scheduling, number and types of resources, how those resources are shared and bound to the algorithm operators, type of data encoding, controller design, floor plan and clock tree design.

The optimal power-consuming algorithm is transformed into hardware. This transformation consists of a number of complex decisions involving scheduling, allocation and binding. Scheduling determines the clock cycle during which an operation is executed; allocation determines the type and the number of resources to be used; binding is the mapping of operations onto resources determined during allocation. Furthermore, resources can be distinguished not only by their function, but also by their internal architecture. For instance, an adder can be realized as a carry-ripple or a carry-select adder.

Figure 7 shows a scheduling graph in ORINOCO. In this graph, the designer is able to see for a particular process, the functional units used in an implementation (multipliers, adders and subtractors in this case), the cycles in which the units are active—including multi-cycle use of the multipliers—and the relationships between these units.

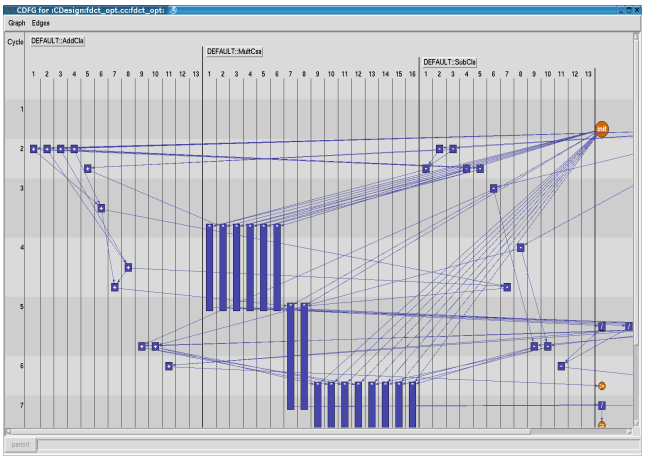

Figure 7. A Scheduling Graph in ORINOCO

This is clearly a large design space with multiple degrees of freedom, each with its own trade-offs and power consumption characteristics. Bounding this space is effected by decomposing the specification into a series of function calls, each of which may be seen as pure function views of subtasks in a complex computation.

Memories are utilized for both intermediate information storage and inter-block communication. Thus, they have a significant effect on chip power consumption, and sometimes account for the majority of it – up to 80% in some SOC designs. Consequently, optimizing memory hierarchy and structure as early as possible is a major step in meeting power consumption constraints.

Common techniques for optimizing memory access and memory system performance include basic loop transformations such as loop interchange, loop tiling, and loop unrolling; array contraction; scalar replacement; and code co-location. Most of these techniques can be affected simply by rewriting code.

Selection of the best optimizations is facilitated by the visual display of power analysis results, as shown in figure 8. The graphic shows an analysis of the power consumption of algorithms used in a digital signal processing application – a Wavelet (signal compression) transform. The bar

graph shows the power consumed, while the memory access traces show memory usage. It can be seen that intra-array optimization reduces power consumption from 19.2µWs to 12.1µWs, or 37%. Inter-array optimization – memory size reduction by mapping arrays onto the same addresses of another array – reduces the consumption by another 1.2µWs, yielding a total 43% reduction.

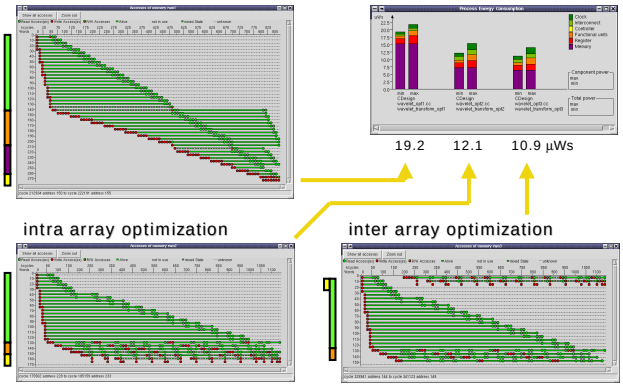

Figure 8: ORINOCO Power Analysis of A Wavelet Transform Mapped To Different Memories

5. Who Are ORINOCO'S Users?

The above discussion leaves open the question as to what sort of designer would actually use a product like ORINOCO that gives power analysis— analysis of what is very much a physical effect--at the ESL, where physical effects have traditionally been an afterthought at best.

The answer really depends on the nature of the organization into which ORINOCO is introduced. Is there a desire to change the situation to which we referred at the end of the last paragraph or, rather, a desire to cope with it.

To see the situation more starkly, consider that one sure way to start a brawl at the holiday party of an electronics party is to get some "back end", i.e., implementation, engineers together with "front end", i.e., system-level architects and ask the former group what they think about the designs that are "thrown over the wall". The back end people will typically roll out a list of horror stories about designs that architects have given them that would require a rollback of the laws of nature to implement. The front end people will mumble something about "synthesis jocks", and note that their mathematically elegant algorithms are the "secret sauce" that makes the finished products so successful. The discussion can only go downhill from there.

There are several points that have to drawn out of this fanciful inter-group "discussion". First, system-level designers typically do not understand physical effects. They are often trained as mathematicians or pure computer scientists (as opposed to those CS people who get their hands dirty in HW-related issues). For such people providing information regarding the physical implications, e.g., the power usage, of their designs may be not very useful—even a child's text in Hindi is incomprehensible to someone who does know Hindi.

Moreover, even if system level designers do have the background and interest in understanding, for example, the power implications of their algorithms, it has not traditionally been very easy for them to discern those implications. Before an implementation of the algorithm exists, they can at best guess based on experience or use crude spreadsheet-based techniques. After implementation, on the other hand, physical information about that implementation is usually so detailed that it is unrealistic to think that many higher-level designers could understand it.

On the other side of the front-end/back-end divide, it should be pointed out that the RTL designers who are given algorithms for implementation generally are neither expert in either algorithm development or in lower-level physical analysis. Thus, their challenge is to take a complex algorithm, understand it, and convert it into an RTL design keeping things like a power budget clearly in mind. Insofar as they can be given tools that help them analyze the algorithms they have been given and help them explore, for example, the power ramifications of various architectures, the better.

This leads us to answer the question as to whom ORINOCO is targeted. In organizations where there is a desire to have system-level designers take a bigger role in vetting their algorithmic designs with regard to power consumption, ORINOCO can be of great use, since it can provide power-related information in a easy to understand manner to algorithm developers. It still requires such designers to have the ability and desire to interpret the information it provides, but it avoids deluging the designer with counter-productive reams of lower-level information.

On the other hand, in organizations where system-level designers will remain agnostic with regard to physics, ORINOCO can be productively used by the consumers of the C/SystemC algorithms, rather than the producers. In such cases, the detailed reports provided by ORINOCO can be used to both analyze the structure of the algorithms to be implemented and to explore various energy efficient architectures of those algorithms and to request changes to those algorithms when what has been designed cannot be implemented efficiently from a power standpoint.

Of course, there will be mixtures of these two organizations. In the ideal, ORINOCO can be used both system-level designers and the designers at the front end of the design flow who consume the system-level design. Indeed, in the best case, ORINOCO can serve as a method of communication between the two groups to make sure that what is implemented has been properly analyzed with regard to power consumption.

Conclusion

An ESL design methodology supported by the appropriate automation tools is the fastest and most effective method of designing complex chips for lower power. Moreover, it significantly reduces the risk of not meeting (often stringent) power constraints by the early identification of function-level hot spots, and enabling the analysis and selection of alternative solutions. ORINOCO, which was introduced in this paper, is ChipVision's implementation of such an ESL methodology.

Dr. Stanley Krolikoski is CEO of ChipVision Design Systems.

Dr. Wolfgang Nebel is Chairman, Chief Technology Advisor and Co-Founder, ChipVision Design Systems and Professor of Computer Engineering at Oldenburg University.

Dr. Laila Kabous is Director of Marketing with ChipVision Design Systems