Design for Low-Power at the Electronic System Level

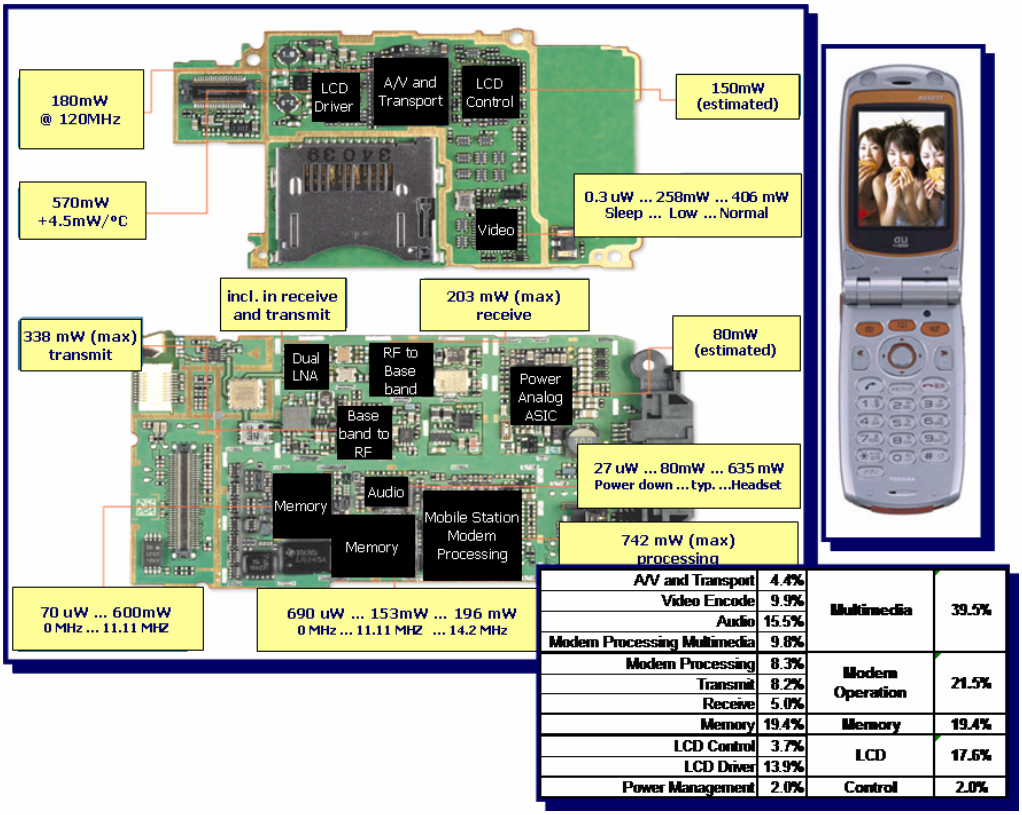
Frank Schirrmeister
ChipVision Design Systems
franks@ChipVision.com

1. Introduction

1.1. Motivation

Well, it happened again. Just when you were about to beat the high score of your favorite game your portable game console powers down because of an empty battery! Design for low-power has become a crucial issue for modern consumer products like portable game consoles, mobile phones and video receivers, MP3 players and multimedia personal digital assistants (PDA). "Standby Time" can easily become the differentiating product feature for a mobile consumer product. But design for low-power is much more far reaching then in mobile applications. Even in wall outlet operated devices like video recorders, DVD players and computers the energy consumption has a profound impact on the cost of the end product. With less power consumption design teams can choose more basic, less expensive packaging and cooling mechanisms.

This white paper will explore the typical distribution of energy consumption in a typical consumer device and introduce the main sources for power consumption We will review different abstraction levels at which energy consumption can be measured or estimated and then introduce common techniques to optimize a design for low power at the pre-RT Level. We will present measurements of the impact these techniques can have on energy consumption using the example of a JPEG Decoder. We will close with a brief summary and assessment which accuracy of energy estimation can be achieved at the pre RT Level



A/V and Transport	4.4%	Multimedia	39.5%
Video Encode	9.9%		
Audio	15.5%		
Modem Processing Multimedia	9.8%		
Modem Processing	8.3%	Modem Operation	21.5%
Transmit	8.2%		
Receive	5.0%		
Memory	19.4%	Memory	19.4%
LCD Control	3.7%	LCD	17.6%
LCD Driver	13.9%		
Power Management	2.0%	Control	2.0%

Figure 1 – Typical Energy Distribution in a Multimedia Mobile Phone
Component Diagram Source: EETimes, February 2nd 2004, David Carey, Portelligent
Sources for Energy Consumption: Publicly available data sheets and ChipVision estimates

1.2. Typical Energy Distribution

Today's mobile phones integrate a variety of different features in a single device. Figure 1 shows a typical phone with picture and video functionality using the built in camera useful for everything from taking still pictures to attending video conferences. The latest TV news can be watched as well as video and audio signals can be digitized, encoded and decoded. A built in MP3 decoder guarantees un-interrupted media availability between phone calls.

Starting from a component list (Source: Teardown Report, EETimes, February 2nd 2004) we analyzed the energy consumptions of the various semiconductor components based on publicly available information and estimates. The operation and control of the LC Display contributes with over 17% of the overall energy consumption. The actual send and receive functionality including the modem processing takes up over 21% while the integrated memories contribute with over 19% to the energy consumption. The enormous portion of multimedia functionality contributes almost 40% to the overall energy consumption (including driving the audio signal to the headset).

This calculation assumes as a worst case that all functionality is used all the time. In reality an application like this will have different use scenarios in which only parts of the phone are used in combination.

In summary the functions with significant algorithmic content (data-path) combined with memory accesses are the main contributors to energy consumption of the semiconductor components in this example.

1.3. Typical Sources of Energy Consumption

The energy consumption in integrated circuits has three main components.

- P_{short}: Energy consumption during the switching of CMOS gates when the complementary parts are open simultaneously. Shorter transition times can help optimize this type of energy consumption.
- $P_{leakage}$: Energy consumption caused by currents during the non-conducting state of gates. Parameters influencing leakage are the supply voltage, the transistor threshold voltage, transistor dimensions like width and length, temperature, IR drop effects, manufacturing tolerances and the state of the gate. While in less aggressive technologies above 180nm leakage had only a small contribution to the overall energy consumption it is a significant contributor at smaller technology nodes.
- $P_{dynamic}$: Dynamic energy consumption happens during the data dependent switching of capacities in transistors and the connections between them. It is typically the largest contributor to the overall energy consumption in a System on Chip (SoC) and has a squared dependency on the supply voltage. Other parameters influencing the dynamic energy consumptions are the switching activity and the size of the capacities to be switched.

  Figure 2 shows as an example the data at the inputs of a multiplier. Whenever a signal changes from 0 to 1 or vice versa capacities have to be switched resulting in charge currents. Dynamic energy consumption can be influenced significantly if correlation between subsequent data values can be optimized to minimize the required switching.

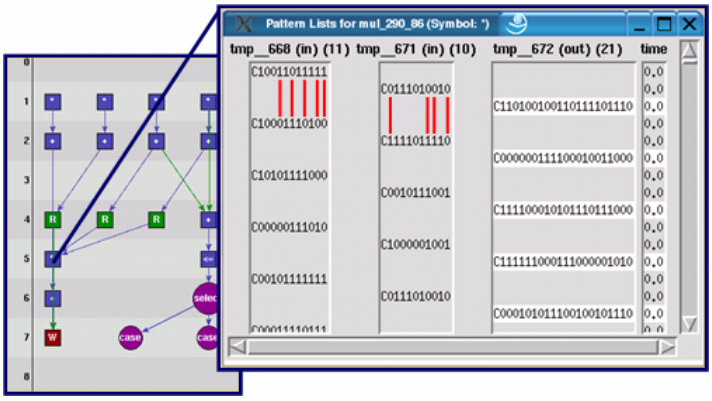


Figure 2 – Activity at the inputs of a multiplier

2. Estimation and Optimization of Energy Consumption

Today design teams have access to a variety of methodologies and development tools to estimate and optimize the energy consumption prior to implementation and availability of silicon for real measurements. Depending on when they are used during the project and at which abstraction level they are applied, tools offer different optimization impact and accuracy.

2.1. Layout Level

Once design teams have reached the actual layout phase which leads to the delivery of a GDSII tape for actual implementation, sufficient data is available to allow accurate simulation and estimation of energy consumption. Depending on the estimates at this level of abstraction design teams can still do transistor sizing and layout re-arrangements to achieve optimization based on placement and interconnect. The accuracy at this level of abstraction is very high but the amount of data to be processed and the simulation speed are so limited that typically only a small number of test vectors can be used for analysis. In addition the leverage on energy consumption is comparatively low as no major design changes like adjustments of the number of arithmetic resources can be made anymore.

2.2. Gate Level

After logic synthesis the gate level netlist can be simulated with event based models. Dynamic energy consumption can be determined fairly accurately based on the design activity. At this level design teams can utilize optimizations minimizing the capacities which have to be driven by the most active nodes in the design. In addition energy consumption can be optimized using balancing of path delays to avoid spikes and spurious transitions and re-timing.

The accuracy of the estimation is still quite high at this level of abstraction but similar to the layout level the amount of data to be dealt with is still limiting. Even if the gate level simulation is performed without back-annotated timing, users have to take into account significant simulation times.

2.3. Register Transfer Level

At the RT Level – prior to logic synthesis – energy consumption can be estimated using event based or probabilistic simulation. In contrast to the gate level the models used at the RT level are more abstract. Other development tools at this level perform a “quick logic synthesis” followed by gate-level estimation.

At the RT Level the number of options to reduce the energy consumption is still quite high, allowing reasonable overall leverage. Specific areas of the design can be dynamically switched to low-power modes trading performance vs. energy during execution. Popular methods are reduced clocking or full clock gating of areas in the design. Optimized resource sharing, isolation of operands and optimized coding of controller and bus states can contribute to reduce the capacities to be switched.

In principle architecture changes are still possible at this level. For example a user may change the number of multipliers to reduce switching via utilization of correlation between subsequent data in a data stream. However, users often focus completely on functional verification during this phase. In addition simulation times are still significant and the actual effort to design the RTL often prohibits focus on energy optimization at this level of abstraction.

2.4. Electronic System Level

Before even arriving at the RT level designers make several decisions, which significantly influence the power consumption of the design. *Micro Architects,* who take an algorithm defined in C or SystemC and decide how to implement it, are facing various high impact areas for low-power optimization. The following sections will introduce typical pre RT level low-power optimization steps, which will afterwards be detailed using the example of a JPEG Decoder in chapter 0.

2.4.1. Bit Width Selection

Typical algorithms defined in C or SystemC will initially not contain definitions of the actual bit width for operations and storage elements. For algorithm selection the design team often relies on floating point and straight integer calculations. Based on the stimulus which is applied to the *Design under Optimization* users can assess the minimum and maximum values on specific operations and then choose the optimal bit width accordingly This allows users to understand the impact of bit width on energy and is a step towards trade offs between *quality*, which may be higher in a video application using higher bit width, vs. *energy* which decreases with lower bit width in the operators.

2.4.2. Trading Performance and Voltage vs. Energy Consumption

Today designers are used to making trade offs between area and performance of an implementation. Increasing the frequency at which an algorithm runs often will increase the energy consumption even though the algorithm could run at lower frequency and still would meet application requirements. The impact of lowering the frequency of an algorithm has significant impact as more operations can be chained between registers hence reducing the overall storage requirements. Alternatively the supply voltage can potentially be reduced resulting in slower performance but significantly reducing energy consumption because voltage contributes to energy consumption in a quadratic way (see 1.3).

2.4.3. Optimizing Design Activity

If the activity of a design can be determined via simulation (see also Figure 2), users can observe the activity profile at the inputs of operations. This information helps to guide optimizations reducing the activity of operations both prior to and after allocation and binding of resources. If the simulation of activity is done at the C/SystemC level it is orders of magnitude faster than signal level simulation at the RT Level, which is commonly used today to create signal level activity files for RT Level power estimation.

2.4.4. Trading Area vs. Energy Consumption

For the micro architecture of the implementation of a given algorithm the design space may allow to implement a schedule using different numbers of resources. The options may be to use 1, 2, 3 or 4 multipliers (as indicated in **Error! Reference source not found.**). Often additional resources will result in less overall energy consumption because with more resources the correlation of the subsequent data in a stream can be used to minimize switching of the capacities at the inputs.

After going though this process the *Micro Architect* can clearly define the optimal number of resources trading area against low-power considerations and articulate this information to the implementation team as part of the *Micro Architecture Specification.*

2.4.5. Memory Optimization

As part of the architecture mapping designers will map arrays in the C Code to different types of memories accessible by the data path. Using memory access trace graphs users can identify areas of suspiciously high number of memory accesses. Users can then consider alternative memory access protocols and their impact on energy consumption.

2.4.6. Clock Gating and voltage scaling

If users understand on a cycle by cycle basis which resources are active and which resources are idle, the modules best suited for clock gating can be identified. Detailed knowledge which modules can be run at lower clock frequencies identifies potential for voltage scaling

	Impact on Energy	Time to Model	Accuracy	Amount of Data	Simulation Speed
Pre-RTL	biggest	short (from C or SystemC)	ok	ok	ok
RT Level	ok	long	ok to good	large	slow
Gate Level	small	longer	good	larger	slower
Layout	smallest	longest	best	enormous	slowest

Figure 3 – Trade offs between different abstraction levels

2.5. Summary - Choosing the right abstraction level for power optimization!

The table in Figure 3 compares advantages and disadvantages of power analysis and optimization at different abstraction levels. It is clear, that the most leverage on energy consumption can be achieved in the pre-RTL design phases. In contrast to

traditional design flows the accuracy at the pre-RT Level may be lower but this is more than offset by that large leverage on design decisions prior to RTL.

3. *Design Example – A JPEG Decoder*

In the following sections we will use the design of a JPEG Decoder to illustrate pre-RTL methods for low-power optimization. The basic structure of the design is outlined in Figure 4.

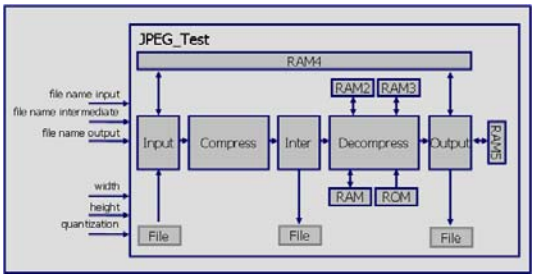


Figure 4 – JPEG Decoder Structure

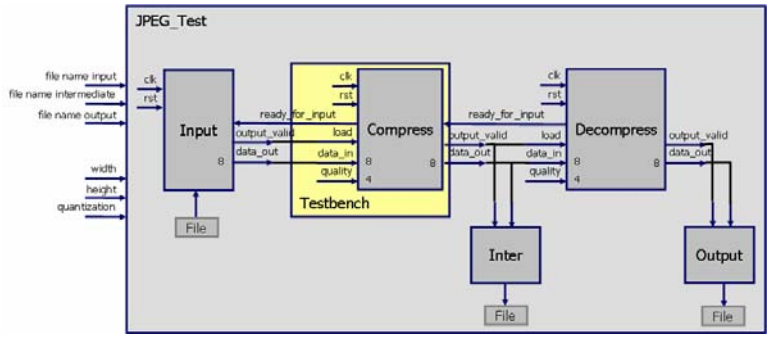


Figure 5 – SystemC overview of the example design

3.1. *Introducing the Design Example*

The design itself has been implemented in SystemC. Figure 5 shows the structure and interfaces of the SystemC modules. The input data are read from a file representing a regression suite of JPEG pictures. The compression itself has been integrated into the design as test bench. The module “Decompress” is the “Design under Development” for which optimization techniques are applied, together with the module reading in data (“Input”), storing intermediate data (“Inter”) and presenting the data at the output (“Output”).

In this particular example the interfaces are representing the protocol of transmission accurately with a request (“ready_for_input”) and a following defined sequence of data transmission on direct connection with an associated data_valid signal. Alternatively transaction level interfaces (like defined in the OSCI TLM working group) could e used. In contrast the content of the modules is coded at a behavioral level, i.e. is using “FOR” loops. Figure 6 shows on the left side a source code example of a matrix multiplication. The corresponding control-dataflow graph (CDFG) is shown on the right side. This graph shows all arithmetic operations, dependencies and memory accesses. The task at hand for the designer is to find the right set of resources to map and bind the behavior to in order to achieve a low-power implementation.

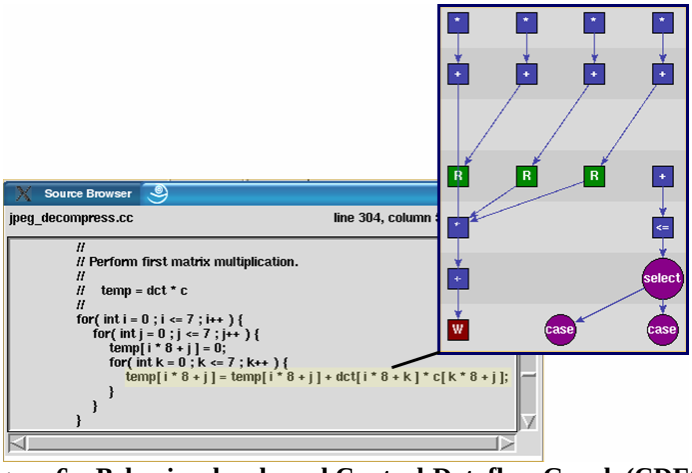


Figure 6 – Behavioral code and Control-Dataflow Graph (CDFG)

3.2. Energy Optimization

3.2.1. Initial Enery Consumption without Optimization

Using ChipVision ORINOCO® technology we estimated the energy consumption prior to any optimization. The results are show in Figure 7. ORINOCO® assumes a macro based standard cell design flow. It bridges the gap between system level design in C or SystemC using a combination of micro architecture prediction, modeling of structural blocks in the design and simulation. The objective is to outline to the user all aspects of energy consumption in a data-flow dominated hardware block and assist the user in optimization.

Figure 7 shows on the horizontal axis the different blocks in the design. On the vertical axis the energy consumption is shown and split into the different components Clock, Interconnect, Local Interconnect, Controller, Functional Units, Register and Memories. The left and right columns per block indicate the best and worst case energy consumptions within the defined constraint space. They represent different target micro-architectures.

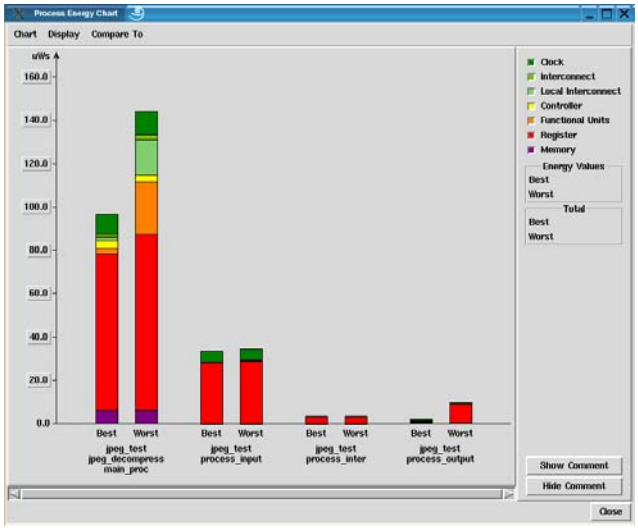


Figure 7 – Initial Estimation of Power Consumption prior to Optimization

3.2.2. Impact of Clock Gating

Figure 7 clearly showed that a significant part of the energy is consumed within the registers of the data-path. Clock gating is a commonly used technique at the RT Level of abstraction. Even at the pre-RT level it is possible to estimate the *possible* impact of clock gating when using different register models for standard and clock-gated registers. From SystemC simulation it is possible to derive in which cycles each register is active to determine when to apply clock gating. Figure 8 shows the result of such an estimate, specifically the saving per module on the left and the new resulting overall energy distribution on the right. It is from this initial estimate obvious that the focus of efforts should be on the actual “decompress” module as it has the highest contribution. In this particular example in excess of 50% of energy can be saved assuming *ideal* clock gating.

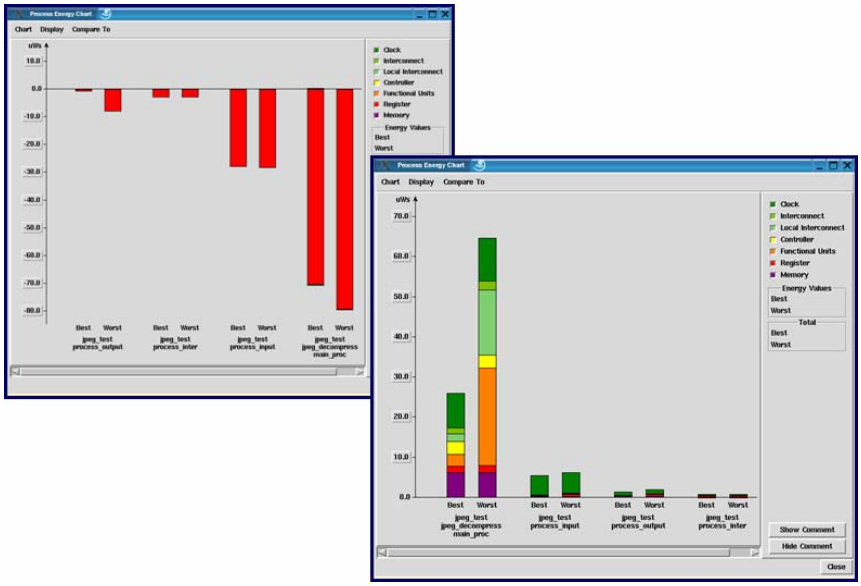


Figure 8 – Impact of Clock Gating

3.2.3. Memory Optimization

Another significant contribution to the energy consumption is caused by the memories in the design. Figure 9 shows on the left side the actual impact of the different memories used in the design. The graph on the right side represents an access trace for one of the key memories, RAM3, which contributes the majority of the energy consumption.

This access trace is caused by the code in Figure 6. It shows the timeline on the horizontal axis and the individual memory cells on the vertical axis. This in essence represents the "lifetime" of each memory cell by clearly marking when a cell is written to and when a value is read out. The experienced user will recognize a suspicious sequence of memory accesses following each other closely, always to the same address. A more detailed analysis of the code reveals that the matrix multiplication has been implemented in an n-optimized fashion. The access to a temporary storage array at temp[i*8+j] is done 8 times even though the surrounding loop does change neither i nor j.

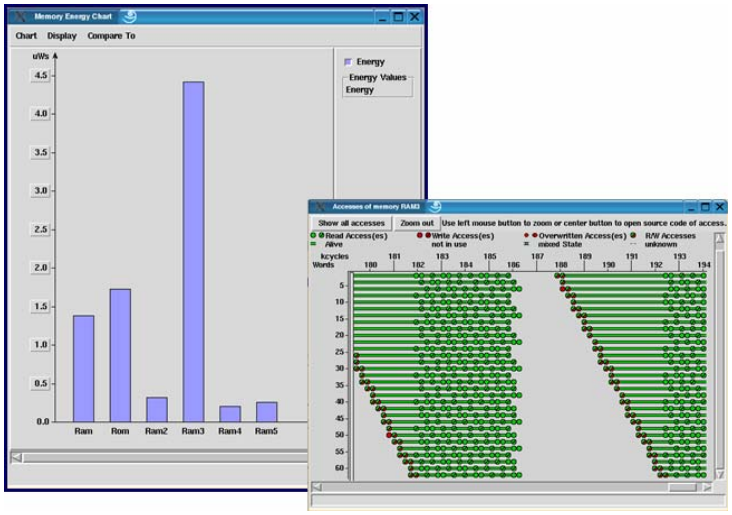


Figure 9- Energy Consumption in Memories and Memory Cell Lifetime/ Access trace

This behavior causes unnecessary memory accesses. Figure 10 shows modified source code which avoids this behavior by using a temporary variable instead of writing back to the memory in each cycle of the loop. The resulting saving in energy consumption is 60% in this example.

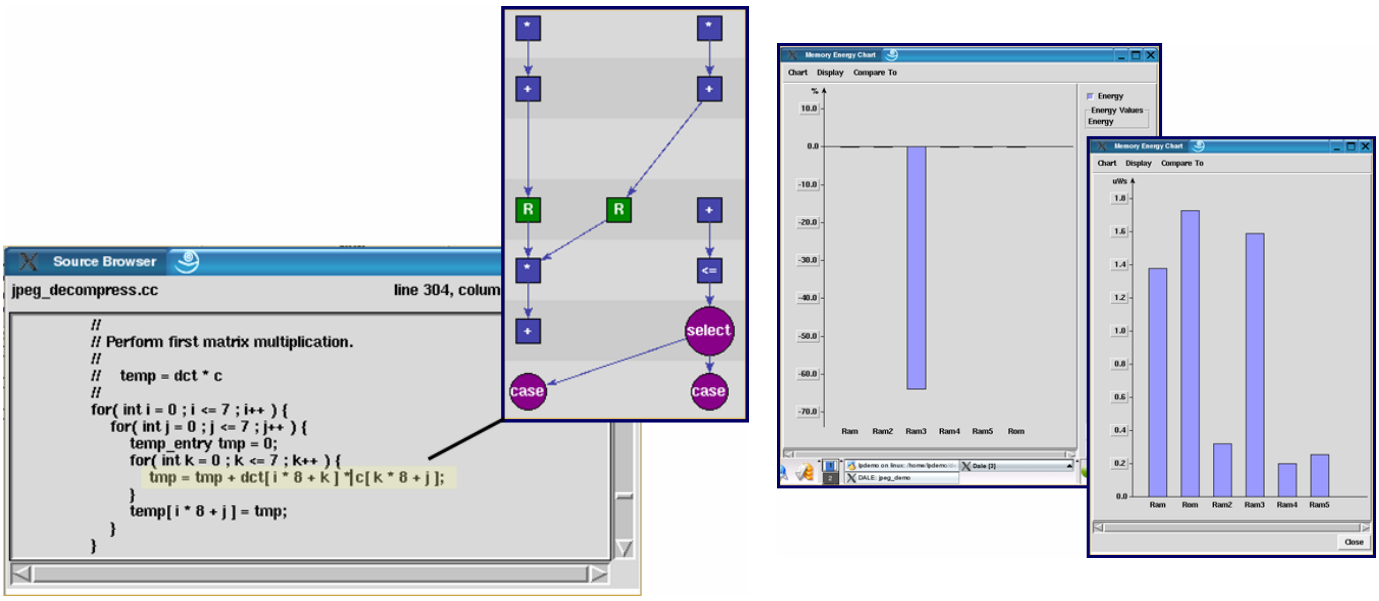


Figure 10 – Optimized Sequence of Memory Accesses

3.2.4. Data Dependent Optimization of Resources

Arithmetic operations like multipliers and adders are a third major contributor to energy consumption. Using SystemC simulation one can create activity traces representing typical use cases for the design under optimization. These activity traces represent the switching at the inputs of the arithmetic components – how many bits of an 8x8 multiplier change from 0 to1 or 1 to 0 per cycle. This in exchange corresponds to the effort of switching the input capacities of these components – the dynamic energy portion in a design.

In design for low-power more can be less – by adding hardware resources, like additional multipliers, the correlation between subsequent data at the inputs can be utilized and the overall energy consumption of the design can be reduced. Figure 11 shows the influence of additional multipliers in our JPEG Design on the energy consumption. When comparing the options to implement a part of the decoder with 1, 2, 3 or 4 multipliers a reduction in energy consumption of more that 60% can be achieved! Even though 3 or 4 multipliers technically reduce the energy consumption even further for a small amount, it is probably not practical to do so as the saving has to be paid with by additional area being used.

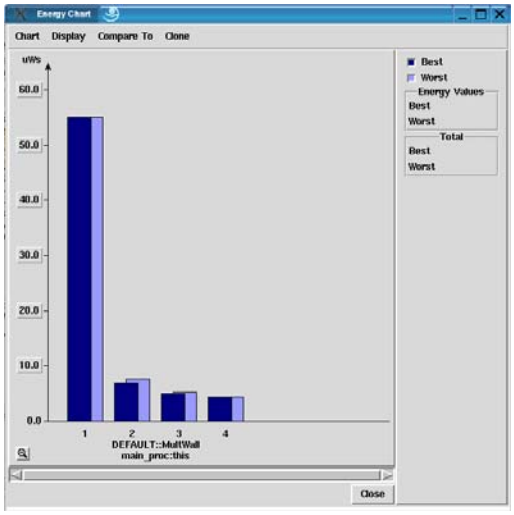


Figure 11 – Influence of Additional Resources on the Energy Consumption

3.3. Accuracy

As mentioned earlier the energy estimation in the pre-RTL phase is less accurate compared to the gate level and layout level. Hence often the primary objective is to make sure that relative assessments of energy consumption between modules and implementation options are consistent and are relatively correct in comparison to their RTL implementations.

However, even in absolute energy estimates the decrease in accuracy is small compared to the increase of optimization potential when raising the level of abstraction as described in 3. Figure 12 shows the comparison of the pre-RT accuracy with layout level estimation as the reference for the example of a Fast Discrete Cosine Transformation. In this case the RTL design has been implemented exactly as defined in the system level assumptions. The overall deviation of the estimates compared to an analysis of the layout data is below 10% while in contrast the optimizations outlined in 3 led to a reduction of energy consumption of 70%. These data are dependent on the application used.

	System-level		Layout		Absolute Deviation
Registers	79.1 nWs	73%	88.1 nWs	78%	-10%
Multipliers	20.6 nWs	19%	13.5 nWs	12%	53%
Global Interconnect	3.4 nWs	3%	3.7 nWs	3%	-8%
Subtractors	1.7 nWs	2%	2.7 nWs	2%	-37%
Adders	1.6 nWs	1%	2.2 nWs	2%	-27%
Controller	1.7 nWs	2%	2.7 nWs	2%	-37%
	108.1 nWs		113 nWs		

Note:
Preliminary Results
Impact of clock tree to be confirmed and not shown here

Figure 12 – Accuracy for a FDCT – pre RT Level vs. Layout

4. Summary

The use of system level solutions like ChipVision's ORINOCO® early in the design cycle leads to significant energy optimization for data-flow dominated blocks and associated memories. Both are significant contributors of the overall energy consumption in a SoC. Our examples have shown energy reduction of up to 75% when applying system level technology.

In addition to the energy optimization potential system level methodologies enable efficient interaction within the design team during the phase in which algorithms specified in C or SystemC are implemented in RTL. Specifically the interaction between the SoC *Micro Architect*, who owns and guides the architecture definition of a data-flow dominated block in a SoC, and the *RTL Designer* implementing the block can be optimized. Using system level methodologies as introduced above the SoC *Micro Architect* can efficiently specify the low-power optimized micro architecture and articulate it to the implementation teams.