ITM Web of Conferences **1**, 02001 (2013)
DOI: 10.1051/itmconf/20130102001
© Owned by the authors, published by EDP Sciences, 2013

Linking Activity to Information and Energy in Hardware

Bernard P. Zeigler[1], Doohwon Kim[1], Chungman Seo[1], Tim Pifer [2] and Roman Lysecky[2]

[1]RTSync. Corp, Sierra Vista, AZ

[2]Arizona Center for Integrative Modeling and Simulation (ACIMS),University of Arizona, AZ

Abstract We propose to extend the framework for relating information and energy via activity [1]. to the design of hardware. Pifer et. al. [2] present an approach to DEVS-based Hardware Design, Synthesis, and Power Optimization that exploits activity concepts. Here we summarize this approach with emphasis on the role of activity. We also provide some results from [2] that show the utility of various mechanisms to reduce power consumption based on DEVS and/or activity features.

1 Introduction

A recent paper proposed to link information and energy via the concept of activity [1]. Energy is the general concept that represents the physical cost of action in the real world. Information is the general concept that models how systems decide on, manage, and control their actions. As in Figure 1 a), information and energy are two key concepts whose interaction is well understood in the following common sense manner: On one hand, information processing takes energy, On the other hand, getting that energy requires information processing to find and consume energy-bearing resources. Systems that sustain themselves in the real world must somehow balance these quantities but without a more rigorous formulation of this relationship it is difficult to study this balance in a general way. We need a more formal concept of activity

(Figure 1 b) to enable us to link energy and information.

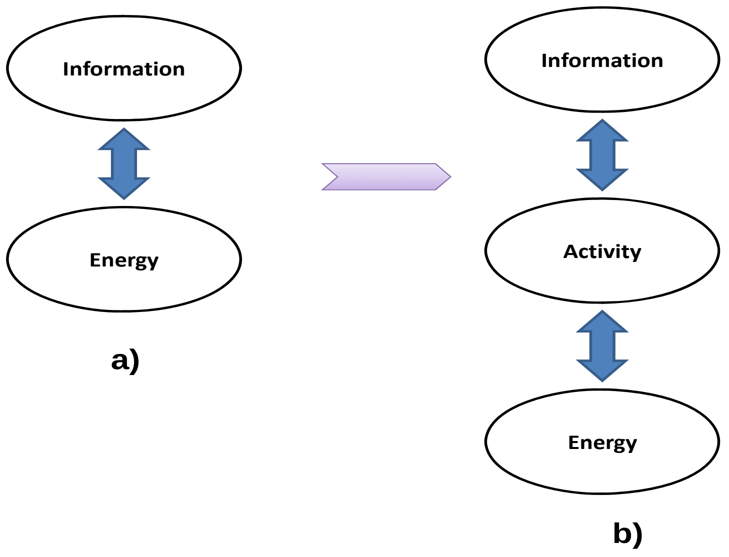


Figure 1. Activity as the concept linking information and energy

Activity is a generic concept (like “information”) refers to the spatial temporal distribution of state transitions in component-based model. Activity concepts have been used to speed up simulation in the form of activity tracking which focuses computational resources on components based on their activity level – it arises naturally in DEVS models with space/time heterogeneity (e.g. crowds, fires.) Just as “information” is a useful abstraction for distinguishing behaviors from physical implementations, “activity” is a useful abstraction to enable energy consumption to be coupled to information flow for a more complete representation of how systems work.

The current approach to design as depicted in Figure 2 introduces resource and energy considerations late in the process.

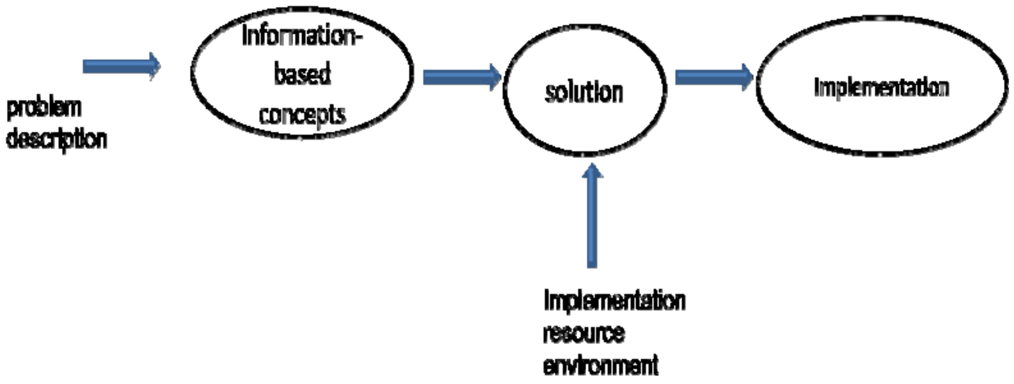

problem description
Information-based concepts
solution
Implementation
Implementation resource environment

Figure 2. Current system design introduces resource environment later in the process

In contrast, the proposed approach in Figure 3 introduces activity-based resource and energy considerations at the start of the process. We propose that the implemented solution will be better because

- activity concepts allow a representation of the resource environment to be exploited earlier in the process,
- the co-dependence of information and activity can be better understood, and
- activity measurement and exploitation can be built in to the implementation architecture to facilitate system development.

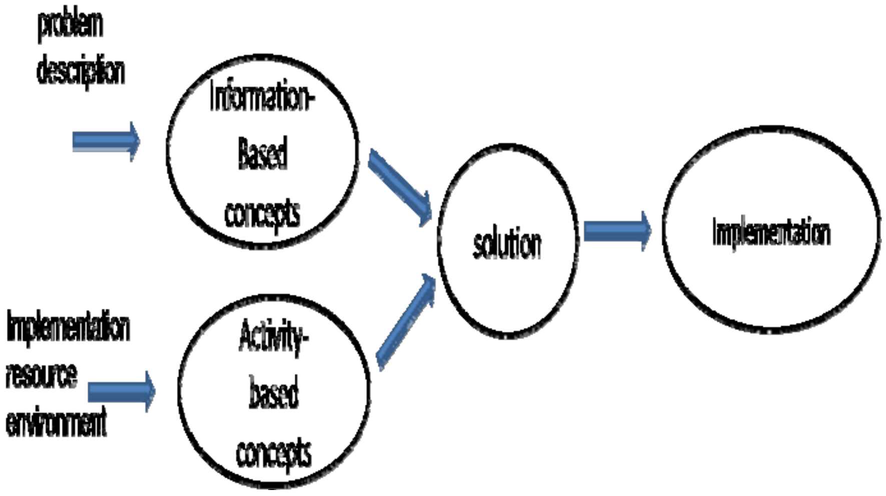
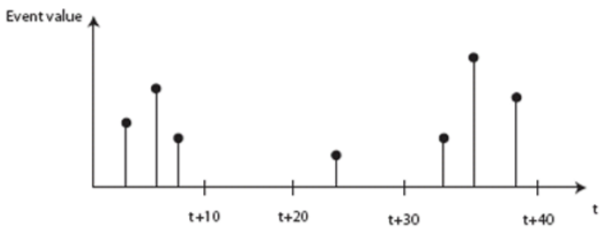


Figure 3. Activity-based system design approach

2 Review of Activity Concepts

One of the unique properties of Discrete Event System Specification (DEVS) [2] is the intrinsic ability of the simulator to be aware of, and therefore, count internal and external state transitions in the model components. Let us measure information processing in a model by such state-to-state transition counts over some time interval, and call this the *activity* measure.



Activity is defined in terms of DEVS model events (internal and external transitions)

$$A(H) = |\{ev_i = (t_i, v_i) \mid 0 < t < H\}|$$

A(H) = activity level in time interval H

A(H) /H = activity rate

Intuitively, components with higher counts over an interval are more actively involved in the information processing than those with low counts. This makes the connection between activity and information. To make the connection with energy, we need to link transition counts with the actual cost of information processing in terms of energy.

In application to DEVS simulation, consider simulating a coupled model with components and couplings. Here activity shows up as calls to the internal and external transition functions, handling of messages, and time coordination. In simulations using event management to handle time coordination, more generally, the activity rate will be reflected in the rate at which event notices are placed into the event list. The greater this rate of event notice processing, the greater the computational resources employed to handle the component sources of these events. Thus in DEVS simulation, the resources accorded to components vary in relation to their activity rates. Since our goal is to relate activity to energy, we will use energy to stand for resources from here on – although most consideration will generalize to resources as well.

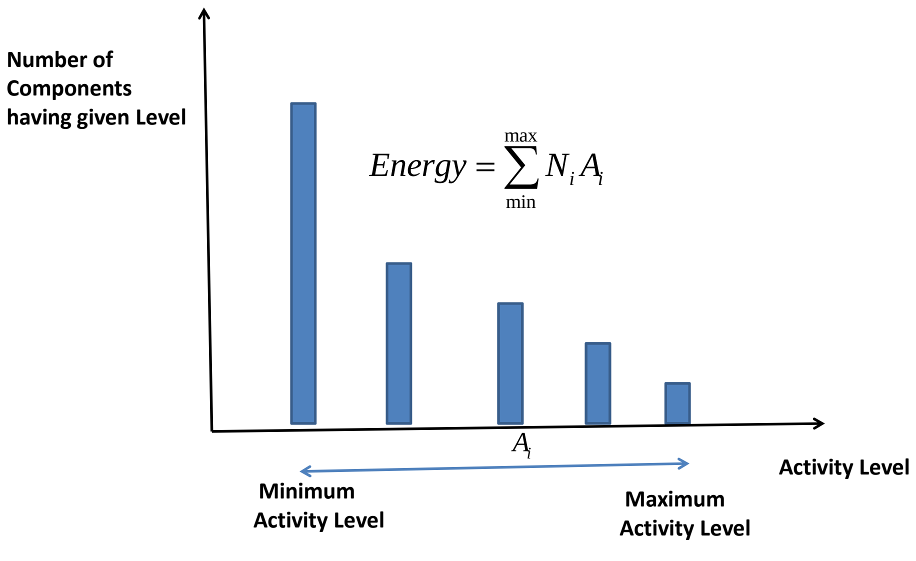


Figure 4. Components having activity level vs activity level

In more detail, consider a distribution of activity rates across components as in Figure 4. If power consumption is directly related to activity rate , or equivalently energy is directly related to activity level, then the “integral” of the distribution proportionately represents the total power consumed.

However, if energy is provided to all at the rate required by the most active components, then the total energy is proportionate to

$$MaxEnergy = N \times A_{\max} \text{ where}$$

$$N = \sum_{\min}^{\max} N_{ii} \text{ is the number of components}$$

The ratio of Energy to MaxEnergy represents the reduction in energy that can be achieved by distributing energy to components in proportion to their activity.

$$\frac{MaxEnergy - Energy}{MaxEnergy} = \frac{A_{\max} - A_{avg}}{A_{\max}}$$

A rough measure of this reduction is the disparity in activity levels, i.e.,

$$ActivityDisparity = \frac{A_{\max} - A_{\min}}{A_{\max}}$$

This ratio is 0 if all components have the same activity level and then there is also obviously no advantage to partition energy according to activity. The closer this ratio approaches unity, the greater the spread in activity levels, the more we can expect to benefit from partitioning energy according to activity.

For the case where there are just two components the energy savings is proportional to disparity:

$$\frac{MaxEnergy - Energy}{MaxEnergy} = ActivityDisparity / 2$$

3 Relating Activity to Hardware

With this as motivation, we discuss a hardware design methodology that opens up the possibility of assigning energy to components in relation to their activity levels. The energy consumed in a hardware FPGA is composed of static and dynamic segments. The static portion is related to the maintenance of the logic itself while the dynamic portion is related to the frequencies of the clocks driving the logic. Traditional hardware design typically uses a single clock and does not pay attention to activity levels so that it risks running at higher than necessary frequency and expending more energy than needed. In contrast, our design methodology can employ multiple clocks and run them at different frequencies allowing the potential to achieve the lower energy levels enabled by activity-based energy partitioning.

Synthesis from DEVS-based model enables design for low-power optimization methods. The basic approach is a globally asynchronous, locally synchronous design pattern that enables efficient clock management and clock gating of individual design elements. Explicitly capturing timing requirements within the system model enables optimization to create a design that differentially assigns clock frequencies. This allows component clocks to run at frequencies that may be much less then would be needed in the standard single clock design.

Three mechanisms to exploit the variation in activity in synthesis of atomic model components are *clock gating*, *handshaking*, and *clock frequency scaling* (differential frequency assignment).

Clock gating circuits remove the clock signal from circuits that are not actively processing. They are controlled by enable and disable signals. The enable signals always have precedence over the disable signals to ensure proper functionality. The disable signal is explicitly determined from the DEVS model passive states. The enable signals are the incoming event requests and the timer signal, which ensure that the clock will be re-enabled when processing is necessary. The timer gate is disabled whenever the logic enters an infinite time advance, and is re-enabled whenever the there is an external event to activate the model.

Handshaking enables communication across different frequencies, which allows each model to be run at the minimum functional frequency, and which reduces the power consumption. Handshaking components require no alteration to hardware description, they work from the event that the model generates then use asynchronous request and acknowledge signal that allows the event to be reliably propagated across different clock domains. Handshaking results in additional latency between events related to the two clock domains that are communicating. Each atomic model includes half of the handshaking hardware for a given event, whether the input event or output event.

Clock Frequency Scaling, The coupled model synthesis is mostly straightforward, following the DEVS coupled model an HDL file instantiates each model and generate the necessary signals to make the connections. This is also the stage that atomic model frequencies are defined. Atomic models are specified in real time, but the hardware implementation uses a discrete timer specified in clock cycles. The number of clock cycles to use is calculated based on the specified time and the specified frequency allowing the same model to run at different frequencies, but maintain the correct behavior.

4 Experimental Test

An Adaptive Quantizer implementation is illustrated in Figure 5:

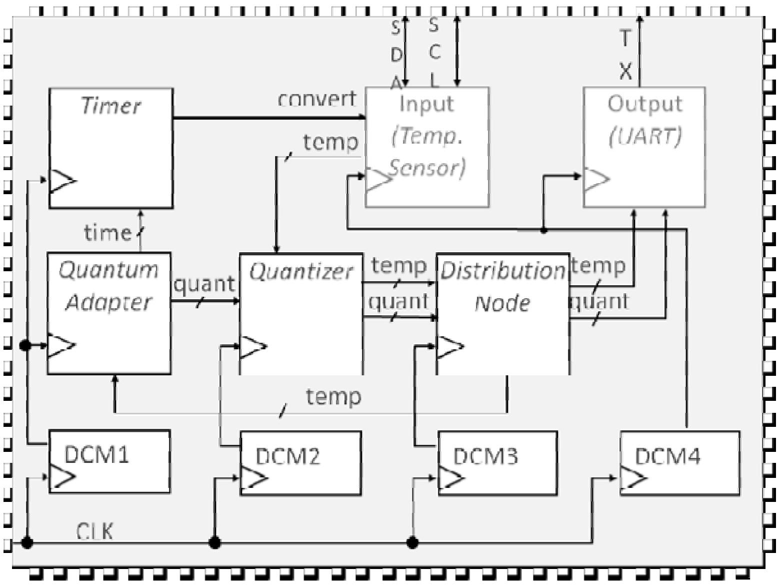


Figure 5. Adaptive Quantizer FPGA Implementation

The components in Figure 5 are:

Quantizer – reads temperature from the sensor and tests if change from last value stored is greater than quantum; if so, it sends the reading to the Distribution node to be output externally

Quantum Adapter – adapts the quantum size - in this experiment on past history. Also generates a duration for the Timer

Timer – times the acquisition of the next input

Input – temperature sensor

Distribution Node – sends sensed and quantized temperature to the Output

Output –UART (universal asynchronous receiver/transmitter) implements the serial port for IBM PCs (so can be connected to an RS-232 interface)

DCM – dynamic clock management module can explore 256 frequencies that are evenly distributed

Figure 6 illustrates the search for assignment of frequencies to meet input/output latency requirements while minimizing energy consumptions.

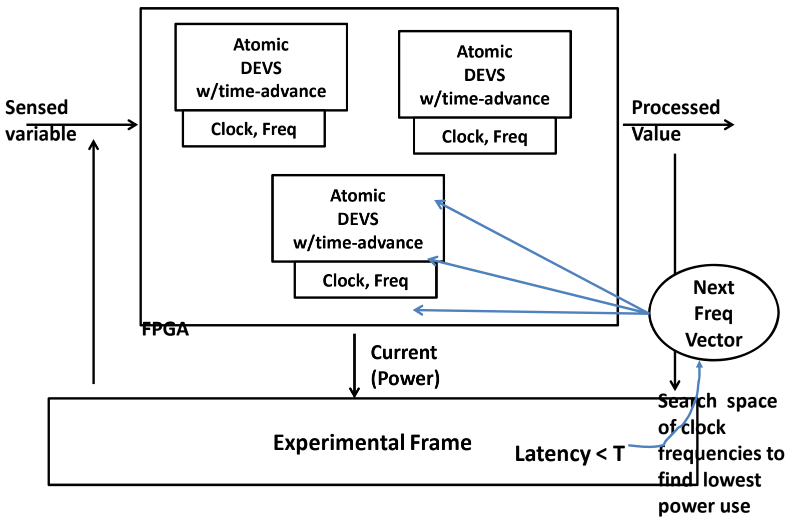

Atomic DEVS w/time-advance
Clock, Freq
Atomic DEVS w/time-advance
Clock, Freq
Atomic DEVS w/time-advance
Clock, Freq
Sensed variable
Processed Value
FPGA
Current (Power)
Next Freq Vector
Experimental Frame
Latency < T
Search space of clock frequencies to find lowest power use

Figure 6. Frequency search to find mimumum power assignment

Test case results are illustrated in Figure 7.

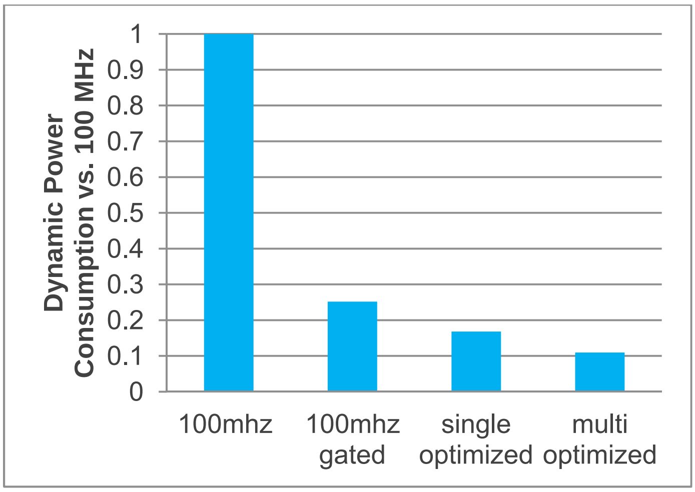


Figure 7. Results for Test Case

Results are shown in relation to power consumption with the native board clock frequency (100mhz.) The addition of clock gating significantly reduced power consumption by approximately 75%. This reduction shows that removing the clock signal from circuits that are not actively processing can be very effective even in the absence of other activity-based mechanisms. Staying with a single frequency, but searching for the frequency that minimizes power, resulting in further reducing power consumption by approximately 30%. When multiple frequencies are used, hand shaking allows frequency scaling to be applied. The experiment showed that another reduction of approximately 35% was achieved in this case.

Results are summarized in the table:

Table 1. Summary of Test Case Results

Exper-iment	**Input/ Output Latency**	**Mechanism**	**Freq-uency Range. MHz**	**Dynamic Power Consumed**
Single Frequency, Native Board Freq-uency	N/A	N/A	100	1
Single Freq -uency, Native Board Freq uency,	N/A	Gated	100	0.25
Single Freq-uency, optim-ized	1 ms	Gated Hand shaking	610	0.18
Multiple Freq-uency, optim ized	1 ms	Gated . Hand shaking	120-720	0.11

A second design experiment was conducted with an application to a medical sensor monitoring and alert generation package. The results confirmed, and indeed improved upon, those shown in the table [3].

5 Summary

We presented activity concepts as a means to bridge the gap between i*nformation-level requirements (behavior and* timing) and energy consumption. This bridge enables system implementations to minimize energy while meeting information-level requirements. In particular, we showed how hardware synthesis from DEVS coupled models can exploit disparity in activities of its components, giving rise to design for low-power optimization methods. The basic approach is a globally asynchronous, locally synchronous design pattern that enables efficient clock management and clock gating of individual design elements. Explicitly capturing timing requirements within the system model enables optimization to create a design that differentially assigns clock frequencies. This allows component clocks to run only when needed and at frequencies that may be much less then would be needed in the standard single clock design.

6 Acknowledgement

This research was supported by RTSync Corp., AZ.

References

1v X. Hu and B. P. Zeigler, “Linking Information and Energy – Activity-based Energy-Aware Information Processing”, SIMULATION April 2013 vol. 89 no. 4 435-450

2v Tim Pifer, Roman Lysecky, Chungman Seo, Bernard Zeigler, DEVS-based Hardware Design, Synthesis, and Power Optimization using Explicit Time Specifications and Deterministic Path Based Latency Analysis, submitted to The International Conference on Hardware/Software Codesign and System Synthesis.

3. Tim Pifer, DEVS-Based Hardware Design, Synthesis, and Power Optimization Using Explicit Time Specifications and Deterministic Path-Based Latency, Masters Thesis, University of Arizona, 2012.