Combining Task Scheduling in Power Adaptive Dynamic Reconfigurable System

Hui Dong, Le-Tian Huang, Jun-Shi Wang, and Terrence Mak

Abstract—Supplying the electronic equipment by exploiting ambient energy sources is a hot spot. In order to achieve the match between power supply and demands under the variance of environments at real time, a reconfigurable technique is taken. In this paper, a dynamic power consumption model by using a lookup table as a unit is proposed. Then, we establish a system-level task scheduling model according to the task type. Based on single instruction multiple data (SIMD) architecture which contains a processing system and a control system with a Nios II processor, a practical dynamic reconfigurable system is built. The approach is evaluated on a hardware platform. The test results show that the system can automatically adjust the power consumption in case of external energy input changing. The utilization of the system dynamic power of their portion is from 80.05% to 91.75% during the first task assignment. During the entire processing cycle, the total energy efficiency is 97.67%.

Index Terms—Nios II, power adaptive, reconfiguration, single instruction multiple data (SIMD), task scheduling model.

1. Introduction

As global energy supply is becoming increasingly strained, combining with the advances achieved by the technology of harvesting energy from the environment, developing ambient energy sources has become a hot spot. Using green energy and emerging energy as the supply of an electronic system, such as solar, bio-energy, and thermal energy, has turned into a new trend. At present, the research achievements like thermal switches, mechanical watches are remarkable[1]. Different from the system supplied by conventional batteries which provide stable energy, taking the dynamic variability of the energy collected from the environment into account, the task of constructing a new model of electronic system energy consumption and supply is the primary consideration. Consequently, the concept of adaptability is put forward, which drives the study of adaptive processing system architecture[2] with greater reliability and adaptability.

Under the demand of power adaptability, a reconfigurable system[3],[4] is re-submitted. The processing system is running by means of dynamical reconfiguration to change the structure of the data processing system at real time. It has a great advantage in cost and performance. Dynamic voltage and frequency scaling (DVFS) was used in [5] and [6] to improve energy efficiency. This paper presents an approach to develop power adaptive computing system which can efficiently use the simulative variable power supply.

In this paper, we use the reconfigurable idea to change the processing system architecture at real time on the system level, and combine task-scheduling means to achieve power adaptability. At design-time, an single instruction multiple data (SIMD) structure of the dynamic reconfigurable system is determined, which maximizes the computation speed on a target hardware platform. The clock gating technique[7] is used, which holds enable control of all functional units driven by the clock, and the adaptability and flexibility of the system are improved by means of task scheduling[8]-[10]. Main work lies in proposing a power model of the processing system and presenting a task scheduling model of the control system. While the system is running, the control system obtains the energy supply data and builds task scheduling at real time, and then sends the control instructions to the parallel processing elements array within the processing system which includes clock gating signals and the task type for each processing element.

The main contributions of this paper are listed as:

• A dynamic power consumption model of a processing element is created on a field programmable gate array (FPGA) hardware implementation level. Thus, the lookup table can be calculated as a unit.

Manuscript received August 7, 2012; revised October 17, 2012.
This work was supported by the National Natural Science Foundation of China under Grant No. 61176025 and No. 61006027, the Fundamental Research Funds for the Central Universities under Grant No. ZYGX2012J003, National Laboratory of Analogue Integrated Circuit Grants under Grants No. 9140C0901101002 and No. 9140C0901101003, and New Century Excellent Talents Program under Grant No. NCET-10-0297.

H. Dong, L.-T. Huang, and J.-S. Wang are with the School of Communication and Information Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: hello_pipi@126.com; huanght@uestc.edu.cn; snake1989@126.com).
T. Mak is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hongkong, China. He is also affiliated with Guangzhou Institute of Advanced Technology, Chinese Academy of Sciences, Guangzhou 511458, China (e-mail: stmak@cse.cuhk.edu.hk).

Digital Object Identifier: 10.3969/j.issn.1674-862X.2012.04.003
A task scheduling model is used at real time to determine schemes including clock gating signals and task types, and adjust system power consumption to the variable power supply.

Evaluate the approach on FPGA; the results show that the power efficient is greatly improved by using the proposed system.

### 2. Power Consumption Analysis

First, we present the structure of the dynamic reconfigurable system to analyze power consumption intuitively.

#### 2.1 Dynamic Reconfigurable System

The overall system architecture consists of two parts which are the processing system and control system. The processing system is comprised of processing elements, and the control system mainly contains the Nios II embedded processor. The total dynamic reconfigurable system is based on the SIMD dynamic reconfigurable processor (DRP) architecture, which is shown in Fig. 1.

In Fig. 1, the processing system on the bottom is made up of DRP (the processing element matrix joint of external RAMs) and reconfiguration memory which are connected to the Avalon bus. The Nios II processor and its peripheral memories, which store the task scheduling model, are connected to the Avalon bus as well. The dotted line presents the information interactions.

The dynamic reconfigurable system works as follows. Firstly, the digital signal of the value of energy supply is read into the processor through the programmed input/output (PIO) port. The processor incorporates the total number of tasks to generate the task scheduling program via the task scheduling model, and then sends the address information and scheduling instructions to the corresponding PIO port of a specific processing element through Avalon bus. Finally the operative mode of each processing element is set up. Consequently, dynamic power adjustment to the processing system can be achieved at real time on condition that power consumption matches with the value of external power supply.

#### 2.2 Dynamic Power Consumption Model

The power consumption of a component is given by

\[ P_c = P_{\text{dynamic}} + P_{\text{static}} \]  

where \( P_c \) is power consumption, \( P_{\text{dynamic}} \) presents dynamic power, and \( P_{\text{static}} \) is static power.

Among all the components that constitute the system, only the power consumption of processing element can be regulated and controlled.

By analyzing the power consumption reports obtained from the power consumption analyzing software by the file (.vcd) achieved after pre-simulation, it shows that dynamic power consumption of the arithmetic logical unit (ALU) varies obviously when the processing elements do different kinds of tasks. In order to simplify the power consumption model and highlight the variable part, we decide to focus on building the model of the dynamic power consumption of ALU.

A well known dynamic power consumption model for CMOS circuits is

\[ P_{\text{dynamic}} = \sum_{\text{All-Node}} \alpha V_{dd}^2 C f \]  

where the capacitance \( C \) and voltage \( V \) can be regarded as constants, and operating frequency \( f \) is set by the designer, thus, the analysis is focused on the transformation rate of signal \( \alpha \). In terms of FPGA internal structure implementation, we consider the lookup table (LUT) as a unit to analyze power consumption.

After obtaining the normalized model in (2), we detail the dynamic power consumption and propose a dynamic power consumption model depending on the probabilistic time of using LUT. LUT is used when the input signal changes, so that the time depends on the change rate of the input signal. During the analysis of dynamic power consumption under different operation types, we need to know the interconnection of LUTs on the data path corresponding to different types of tasks, then analyze the sum of the probabilistic time of using each LUT on the critical data path under one period of operation. All the interconnect information can be extracted from Quartus II Technology Map Viewer.

An \( I \)-bits input signal corresponds to \( 2^I \) kinds of states, and the probability of one state is

\[ P_N = \sum_{i} \left( (n_i - 1) z_i + n_i u_i \right) \]
where \( z_i \) and \( u_i \) denote the probability of the input signal \( n_i \) to be 0 or 1, respectively.

Then, the state transition probability from state \( N = (n_1, n_2, \ldots, n_N) \) to state \( M = (m_1, m_2, \ldots, m_M) \) can be calculated by the transformation rate of the input signal:

\[
P_{N,M} = \prod_{i} \left[ n_i - m_i \mid c_i + | n_i - m_i - 1 | (1 - c_i) \right]
\]

where \( c_i \) stands for the transformation rate of the input signal \( n_i \) and the whole state transition probability \( P \) is a matrix of \( 2^N \times 2^M \).

In practice, as the input signals transmit on different paths, the time they arrive differs. Thus, signal burrs should be taken into account. The signals which arrive at the same time are put into a team. The \((N, M)\)th element of the burr-coefficient matrix \( W \) is

\[
W_{N,M} = \sum_{i} \left( (n_i \odot m_i) \mid \cdots \mid (n_{i+h-k} \odot m_{i+h-k}) \right)
\]

where \( W_{N,M} \) presents the state transition probability from state \( N = (n_1, n_2, \ldots, n_N) \) to state \( M = (m_1, m_2, \ldots, m_M) \) and the operator "\( \odot \)" expresses the binary operation of or.

According to the actual hardware implementation structure, LUTs’ arrangement includes cascades. Consequently, it is necessary to consider the parameters of the output signal, including the probability of the output signal to be 0 and 1, and the rate of change of the output signal. They can be expressed as below:

\[
P_{\text{output}=0} = \sum P_N = 1 - P_{\text{output}=1}
\]

\[
P_{\text{output-change}} = \sum_{N,M} \left| \text{output}_N - \text{output}_M \right| P_{N,M} P_N .
\]

In condition, to an LUT, its probabilistic lookup times is

\[
\text{Times} = \sum_{N,M} P_N P_{N,M} W_{N,M} .
\]

We use this model into practical calculation in our processing unit.

### 3. Task Scheduling Model

The task scheduling model mainly completes the real-time scheduling towards the processing elements array by sending instructions and adjusting the operating frequency in order to maximize the utilization of energy. The Nios II Integrated Development Environment (Nios II IDE) platform provides C/C++ programming environment for the designer to establish the task scheduling model. In practice, on the basis of the external simulation value of the energy supply, the control system generates a scheduling scheme to achieve the processing system real-time reconstruction and task scheduling towards the processing system.

### 3.1 Task Scheduling Program

While setting up the task scheduling model, external energy supply and the total number of tasks of each task type are known. We set the highest energy utilization efficiency and the shortest system performing time as the target of the optimization model, aiming to get the task schedule, as shown in Table 1.

In Table 1, \( m_{ij} \) presents the number of processing elements which do the task type \( i \) during task assignment \( j \) that lasts for \( h(j) \) execution cycles. It can be seen from the table that the entire implementation cycle is divided into several task assignments. The scheduling scheme stays the same during one task assignment whereas differs in different task assignments. Thus, between two adjacent task assignments, the control system needs to re-schedule.

The optimization model is as follows

\[
\begin{align*}
& \left\{ \max \left[ \sum_{j=1}^{N} P_j T_{pe} + t(P_{\text{state}} + P_{\text{cpu}}) \right] \right\} \div T_{\text{sup}} \\
& \left\{ P(m) \leq P_{\text{sup}} \right\} \\
& \left\{ 1 \leq m \leq K \right\}
\end{align*}
\]

where \( N \) stands for the number of task types, \( P_i \) and \( l_t \) present the dynamic power consumption of one processing element and the task size of task type \( i \), respectively. \( T_{pe} \) is the operating period of processing unit, \( P_{\text{cpu}} \) is the power consumption of central processing unit (CPU), and \( P_{\text{sup}} \) is the power supply, \( t \) is the operating time of the system. \( K \) is the total number of processing elements, \( m \) is the number of processing units that are turned on. The objective function (9a) is set to achieve the highest energy utilization efficiency, and there are two constraints (9b) which are from the energy supply and the limitation of the hardware resources.

The operating time \( t \) is made up of the task scheduling time of Nios II embedded processor \( t_{\text{prepare}} \) and the processing time of processing element \( t_{\text{process}} \):

\[
I = t_{\text{prepare}} + t_{\text{process}}
\]

\[
t_{\text{process}} = \sum_{j=0}^{w-1} T_{pe} h(j)
\]

\[
t_{\text{prepare}} = \frac{1}{2} \sum_{j=1}^{w-1} \sum_{i=1}^{m_{ij}} |m_{ij} - m_{ij+1}| + \sum_{j=1}^{w-1} \sum_{i=1}^{m_{ij}} |m_{ij} - m_{ij+j+1}| \sum_{j=0}^{w-1} T_{cpu} + \frac{1}{2} \sum_{j=1}^{w-1} m_{ij} T_{cpu}
\]
where \( w \) is the times of tasks assignment and \( T_{cpu} \) is the operating period of CPU.

### 3.2 Software Implementation

The software implementation needs to comply with the requirements of the objective function (9a) while satisfying the limiting conditions. According to (9), as the system architecture and the task size are determined, only the operating time can be optimized. Therefore, software implementation targets on the shortest operating time.

Operating time can be divided into the task scheduling time of Nios II embedded processor \( (t_{\text{prepare}}) \) and the processing time of processing element \( (t_{\text{process}}) \). In order to reduce the time of scheduling time, it needs to decrease the communication time on Avalon bus. From Table 1, we should accomplish two targets:

- Minimize the number of task assignments and make sure that the processing system completes as many tasks as possible in a task assignment.
- Make the schedules of two adjacent task assignments as similar as possible.

Through the analysis above, the specific software implementation process is shown in Table 2, where \( v \) is the level of processing frequency, and \( f_{pe} \) is the operating frequency of processing element.

### 4. Experimental Results

In our experiments, the power adaptive ability of the dynamic reconfigurable system is evaluated on the testing platform shown in Fig. 2.

Specific architecture of the system on FPGA is shown in Fig. 2. The parameters of 8808A multimeter are: measuring range is 10 A, low sampling rate 15 samples/s, and precision 1 mA.

#### Table 2: Task schedule algorithm table

<table>
<thead>
<tr>
<th>Task scheduling algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: input task size ( n ) and power supply ( P_{i} )</td>
</tr>
<tr>
<td>2: for the system start running do</td>
</tr>
<tr>
<td>3: ( \sum n_{i} P_{i} / P_{i} = h' )</td>
</tr>
<tr>
<td>4: ([h'] = h )</td>
</tr>
<tr>
<td>5: if ( h = 0 ) then</td>
</tr>
<tr>
<td>6: ( m_{ij} = n_{i} )</td>
</tr>
<tr>
<td>7: if ( \sum m_{ij} &gt; K ) then</td>
</tr>
<tr>
<td>8: ( v = v + 1 ), ( f_{\text{pe}} = f_{\text{pe}} \times v ), ( P = P_{i} \times v )</td>
</tr>
<tr>
<td>9: else if ( h(j) = 1 ) break</td>
</tr>
<tr>
<td>10: else ( m_{ij} = [n_{i} / h], h(j) = h )</td>
</tr>
<tr>
<td>11: if ( \sum m_{ij} &gt; K ) then</td>
</tr>
<tr>
<td>12: ( v = v + 1 ), ( f_{\text{pe}} = f_{\text{pe}} \times v ), ( P = P_{i} \times v )</td>
</tr>
<tr>
<td>13: else if ( \sum m_{ij} &lt; 3 ) then</td>
</tr>
<tr>
<td>14: ( v = v - 1 ), ( f_{\text{pe}} = f_{\text{pe}} \times v ), ( P = P_{i} \times v )</td>
</tr>
<tr>
<td>15: else ( n_{i} = n_{i} - m_{ij}, j = j + 1 )</td>
</tr>
</tbody>
</table>

Table 3: Cmp-operation data analysis

<table>
<thead>
<tr>
<th>LUT</th>
<th>Probabilistic lookup time</th>
<th>Change rate</th>
<th>( P{\text{output}=0} )</th>
<th>( P{\text{output}=1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LessThan0~0</td>
<td>0.9844</td>
<td>0.2188</td>
<td>0.8750</td>
<td>0.1250</td>
</tr>
<tr>
<td>LessThan0~1</td>
<td>0.9844</td>
<td>0.4922</td>
<td>0.5625</td>
<td>0.4375</td>
</tr>
<tr>
<td>LessThan0~2</td>
<td>0.9375</td>
<td>0.4688</td>
<td>0.6250</td>
<td>0.3750</td>
</tr>
<tr>
<td>LessThan0~3</td>
<td>1.4063</td>
<td>0.4980</td>
<td>0.5313</td>
<td>0.4688</td>
</tr>
<tr>
<td>LessThan0~4</td>
<td>1.8513</td>
<td>0.4894</td>
<td>0.5284</td>
<td>0.4716</td>
</tr>
</tbody>
</table>

* where LessThan0~X is the identifier of each LUT.

In this experiment, we assume that the supply voltage of the FPGA board is constant, and measure the current of the power port to reflect the total power consumption. In general, this measurement cannot be stable current value, in order to get more accurate results, collecting a large number of data and doing the average are needed.

#### 4.1 Power Consumption Model Test

The power consumption model is established based on the specific implementation structure on FPGA as mentioned in Section 2. In this design, the processing unit does three kinds of tasks: AND, ADD, and CMP. (compare). According to the model and practical design, we could get the theoretical analytical solution firstly.

The data path of AND-operation consists of eight LUTs which are arranged in single stage and have parallel outputs. The operands are random and the transformation rate of each input signal is 1/2, so the probabilistic lookup times on the entire data path is 6.

ADD-operation is based on the data path which is made up of 8 LUTs cascaded together, and CMP-operation corresponds to 5 LUTs. These two operations both need to consider the signal burrs. Using Matlab programming, their probabilistic lookup time is got. The results of CMP-operation is shown in Table 3 as an example, which include probabilistic lookup times, change rate of output signal, and probability of the output-signal state.

Summarizing the lookup times of all the LUTs we can get the total probabilistic lookup times of ADD- and CMP-operations which are 9.3335 and 6.1639, respectively.

The control system sends instructions to turn on all the 128 processing elements and do 3 kinds of operations and non-operation. The experimental results are plotted in Fig. 3. The four figures present the measured power and theoretical power (dashed lines) for AND-operation, ADD-operation, CMP-operation, and non-operation from top to bottom.
Table 4 shows the normalization results of measured power, theoretical power, and estimated power. Theoretical power is calculated via the proposed power consumption model, while estimated power is obtained from the power consumption analyzing software. We can tell from the table that the trend comparison among three task types is the same in three computing environments. Compared with the measured power, the result calculated in our power consumption model on task type cmp. has a relative error of 8.77%.

We summarize the reasons of difference existed between the theoretical power and measured power, for example, the randomness of operands, the small proportion that the dynamic power accounts in the whole power consumption. Taking the particularity of the measured results into account, we use the measured normalized dynamic power in the task scheduling model after.

4.2 Task Scheduling Model Test

In this experiment, we change the normalization coefficient of power supply from 50 to 150, and the control system runs the task scheduling model at real time.

At the beginning of the test, the power supply coefficient is set as 50. It can be seen from Fig. 4 that the measured power stays lower than the theoretical power supply, and the three steps in DC during the time steps period from 65 to 150 present a certain appearance that the task schedule includes 3 task assignments, which satisfies the trend of estimated power. When the supply coefficient changes from 50 to 150 at the time of 65, the measured power increases sharply, and the operating frequency of processing system jumps from 50 MHz to 100 MHz automatically at the time of 150.

In fact, only nearly 10% of the total power consumption is taken by the dynamic power, we just calculate the utilization efficiency of the proportion of dynamic power consumption. After removing the data collected during the reset period, the results are as shown in Fig. 5.

Experimental data show that dynamic power utilization efficiency changes from 51.24% to 91.75%. If only considering the first task assignment, in order to ignore the period of dealing with the remaining tasks, the utilization efficiency of dynamic power can stay more than 80.05%.

Table 4: Task schedule table

<table>
<thead>
<tr>
<th>Task type</th>
<th>AND</th>
<th>ADD</th>
<th>CMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theoretical power</td>
<td>0.0870</td>
<td>0.1354</td>
<td>0.0894</td>
</tr>
<tr>
<td>Estimated power</td>
<td>0.0870</td>
<td>0.1152</td>
<td>0.0881</td>
</tr>
<tr>
<td>Measured power</td>
<td>0.0870</td>
<td>0.1154</td>
<td>0.0912</td>
</tr>
</tbody>
</table>

Fig. 4. Measured power, theoretical power, and estimated power for task scheduling model test.

Fig. 5. Dynamic power efficiency in task scheduling model test.
As for the entire running period, the total system energy utilization efficiency is 97.67%.

5. Conclusions

In this paper, we propose a power consumption model and a task scheduling model used in the dynamic reconfigurable system based on SIMD architecture which is applied to energy harvesting environment. Through the task scheduling executed in Nios II processor, the system can automatically adjust the power consumption in case of external power supply changes.

Future work will focus on improving the system architecture, optimizing the design of the power model, and investigating the scheduling methods to achieve practical application in days to come.

References


Hui Dong was born in Liaoning Province, China in 1990. She received the B.S. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu in 2012, in communication engineering. She is currently pursuing the M.S. degree with the Department of Communication and Information Engineering. Her research interests include power adaptive computing and communication, power management.

Le-Tian Huang was born in Sichuan Province, China in 1984. He received the B.S. and M.S. degrees from UESTC, Chengdu in 2006 and 2009, respectively, both in communication and information system. He is a lecturer with UESTC. His research interests include power adaptive computing and communication, FPGA/ASIC design.

Jun-Shi Wang was born in Liaoning Province, China in 1989. He received the B.S. degree from UESTC, Chengdu, in 2012, in communication engineering. He is currently pursuing the Ph. D. degree with the Department of Communication and Information Engineering. His research interests include power adaptive computing and communication, power management.

Terrence Mak received his Ph.D degree in electrical and electronic engineering from Imperial College London, UK in 2009. He is an assistant professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research interests include multi-core and embedded systems design and high-performance bio-computing systems.