IEICE TRANS. INF. & SYST., VOL.E98–D, NO.2 FEBRUARY 2015

PAPER Special Section on Reconfigurable Systems

Energy Efficiency Improvement by Dynamic Reconfiguration for Embedded Systems

Kei KINOSHITA[†], ***Nonmember***, **Yoshiki YAMAGUCHI**[††a)], ***Member***, **Daisuke TAKANO**[†], **Tomoyuki OKAMURA**[†], ***and*** **Tetsuhiko YAO**[†], ***Nonmembers***

SUMMARY This paper seeks to improve power-performance efficiency of embedded systems by the use of dynamic reconfiguration. Programmable logic devices (PLDs) have the competence to optimize the power consumption by the use of partial and/or dynamic reconfiguration. It is a non-exclusive approach, which can use other power-reduction techniques simultaneous, and thus it is applicable to a myriad of systems. The power-performance improvement by dynamic reconfiguration was evaluated through an augmented reality system that translates Japanese into English. It is a wearable and mobile system with a head-mounted display (HMD). In the system, the computing core detects a Japanese word from an input video frame and the translated term will be output to the HMD. It includes various image processing approaches such as pattern recognition and object tracking, and these functions run sequentially. The system does not need to prepare all functions simultaneously, which provides a function by reconfiguration only when it is needed. In other words, by dynamic reconfiguration, the spatiotemporal module-based pipeline can introduce the reduction of its circuit amount and power consumption compared to the naive approach. The approach achieved marked improvements; the computational speed was the same but the power consumption was reduced to around $\frac{1}{6}$.

key words: *dynamic reconfiguration, power performance, video processing, image recognition, object tracking, embedded / wearable computing*

1. Introduction

The reduction of power consumption has become an important problem in not only super computers but also electronic devices [1], [2]. For example, the concept, dark silicon, comes from the gap between Moore's Law [3] and Dennard scaling [4]; the shortage of LSI power disables a larger fraction of circuits whenever process rule advances. The frequency control such as Intel [5], AMD [6], and NVIDIA [7] is proposed as a means of handling the problem. But, the approach is not a fundamental solution because there are a measurable amount of on-power transistors in an idle state, where they do not contribute to any computation but consume their power. Heterogeneous architecture [8]–[10] is one candidate to overcome the problem. Thus, this paper proposes a reduction approach with the use of heterogeneous modules by reconfigurable computing.

Manuscript received May 9, 2014.
Manuscript revised September 10, 2014.
Manuscript publicized November 19, 2014.

[†]The authors are with the Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba-shi, 305–8573 Japan.
[††]The author is with the Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba-shi, 305–8573 Japan.
a) E-mail: yoshiki@cs.tsukuba.ac.jp

DOI: 10.1587/transinf.2014RCP0015

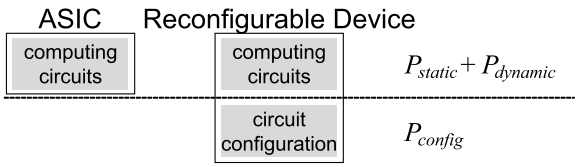

Fig. 1 Electric power composition of ASICs and reconfigurable devices.

This paper starts with the definition of the power consumption of a device, and then discusses its reduction. Figure 1 shows the brief overview of electrical power composition of an ASIC and a reconfigurable device.

As shown in Fig. 1, the power consumption of an ASIC, P_{ASIC}, can be expressed simply as the following equation:

$$P_{ASIC} = P_{static} + P_{dynamic}, \quad (1)$$

where $P_{dynamic}$ and P_{static} are the power consumption of logic circuit activities and leakage currents, respectively. The both P_{static} and $P_{dynamic}$ increase in proportional to the number of transistor gates of a device, namely the size of a device. Toward the power reduction of a device, its size reduction is an effective approach.

The power consumption of a reconfigurable device, P_{RECONF} can be expressed as follows.

$$P_{RECONF} = P_{config} + P_{static} + P_{dynamic}, \quad (2)$$

where P_{config} is required for holding its circuit configuration. P_{config} may negate the beneficial effect of a size reduction but it is an essential piece to hold a circuit design as an electrical data. If a large computing circuit is designed by a small chunk of circuit configuration, it can help to solve this problem. Thus, coarse-grained reconfigurable architecture (CGRA) [11]–[15] has been attracting many researches [16], [17]. This paper also focused on CGRA and proposed a spatiotemporal module-based pipeline to design an energy-efficient system. It balanced the profits and losses, which was evaluated in a CGRA system [18], [19].

An augmented reality (AR) system was chosen for the evaluation. AR has rapidly spread to daily life as typified by mobile applications but, the contradiction, performance and power consumption, has vexed many industries. For example, each system would like to compute 30 frames-per-second (fps) video in VGA (640x480) resolution. On the other hand, each is a battery-powered system and its computational device must consider extreme power saving for

long-term execution. Thus, the evaluation of an AR application in a real system can serve as a test bed for future systems.

Moreover, the proposed approach is evaluated from not only its power-performance efficiency but also the accuracy of implemented proposed algorithms. The system treats pattern recognition and object tracking. The former uses the subspace method [20] and the latter adapts the Kanade-Lucas-Tomasi (KLT) algorithm [21]. These algorithms are widely known in computer vision but the both approaches are unsuited for reconfigurable architectures because they include much complex calculation such as the extraction of a square root of 32-bit or 64-bit data width. Thus, the approaches specialized for reconfigurable device are required for achieving higher performance. This paper provides a fixed-point-and-space-reduced subspace method and an elliptical KLT tracking method. Each one of them is assessed on the real hardware system and the details of the results are described in this paper.

2. Background

2.1 Power Consumption of Interconnect Switches

Reconfigurable devices have not only on-chip memory storage for a circuit configuration but also interconnect switches [22] which connects each element on a device such as I/Os, logics and on-chip RAMs. The interconnect switches are a critical component to configure a circuit design. But it does not represent a direct contribution for the computation.

The clear discussion of the power reduction of a device requires some sort of quantitative evaluation of interconnect switches though it is so difficult. This paper assumes that the ratio of their power consumption shows an insignificant impact on the benefit of reconfigurable computing. The reason is the ratio does not reach 10 percent even if a fine-grained device was selected [23], [24]. Furthermore, a coarse-grained reconfigurable architecture (CGRA) devices can expect to reduce their ratio.

2.2 The Application Range by Dynamic Reconfiguration

The constituents of the power consumption is composed of P_{config}, P_{static}, and $P_{dynamic}$ as shown in Fig. 1.

P_{config} is dependent on the reconfigurable architecture which is adopted to a system. In general, the power consumption of a CGRA device is smaller than that of a fine-grained reconfigurable architecture device.

P_{static} is dependent on the number of power-on transistors. It can be eliminated by a power-gating approach [17], [25] but not a clock-gating approach [26], [27]. Thus, the reduction of the number of power-on transistors, namely a device size reduction, is a vital component in the reduction of P_{static}. The use of dynamic reconfiguration and a functional-based design is helpful as a way of the size reduction.

$P_{dynamic}$ is dependent on logic circuit activities. If all circuits are fully-pipelined and operating, $P_{dynamic}$ is not reduced by any approach because they can not be removed at all. But, if not, reconfigurable computing has the potential to reduce it. For example, supposing that loop computations and functional pipelines are implemented, Fig. 2 gives a typical instance.

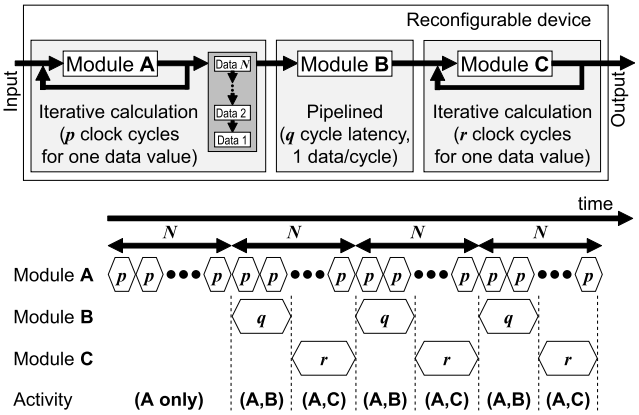

Fig. 2 Circuit activities and computational timelines.

In this figure, `Module A` always operates while an external device is sending a data stream to the reconfigurable device. One loop computation in `Module A` takes p clock cycles in this case. When N output data are required in the next module, it takes $N \times p$ clock cycles in total until the next computation can begin. And then, the computational result of `Module B` is output to `Module C` in q cycles. Finally, `Module C` output the final result in r clock cycles and one series of computations in Fig. 2 ends.

Here, the fact remains that the three modules, `Module A`, `B` and `C` do not work at the same time though we can say all modules are used for the computation. For example, at most two modules run simultaneously in Fig. 2. Thus, if reconfigurable computing follows not only a spatial implementation but also a fine-grained computational timeline, it can reduce the power consumption without any disadvantage. The benefit can be described in Sect. 2.3.

2.3 Power Reduction by Dynamic Reconfiguration

Before discussing the optimization of circuit configurations, three main groups in reconfigurable circuits are described as follows.

Active the circuits in this class mean computational circuits configured by a circuit configuration. In other words, the circuit information is loaded to physical entities including interconnects. It can work as a meaningful computational module. Thus, all the power, P_{config}, P_{static}, and $P_{dynamic}$, is consumed.

Idle the circuits in this class is almost the same as `Active` circuits. The difference is whether the computation on the circuits is effective or not. For example, the activity of circuits may be controlled by enable signals which are generated by other circuits. In another example,

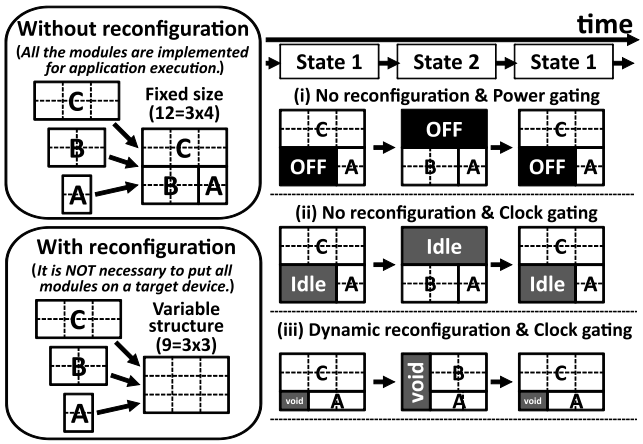

Fig. 3 Power reduction using a dynamic reconfigurable device.

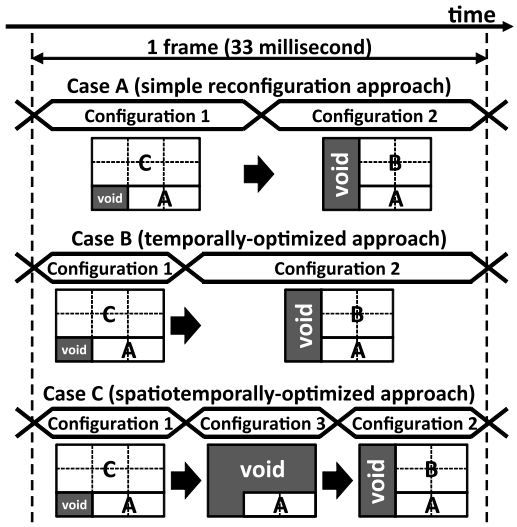

Fig. 4 Difference of clock-gating approaches in dynamic reconfiguration.

circuits may wait input datas in the same manner as pipeline stalls of a processor. The important part is that the wiring network, clock line network, and combination logic are alive though the output data are not updated in these example. Thus, P_{config} and P_{static} are not reduced. $P_{dynamic}$, may be reduced but not removed but the effect is dependent on the combination of device architecture and circuit design.

Void the circuits in this class has no configuration data, and therefore the circuits has a different behaviour to that of `Idle` circuits. That is, it can expect the larger reduction of $P_{dynamic}$ compared with `Idle` circuits. Furthermore, P_{config} and P_{static} may also be reduced if the reconfigurable architecture supports configuration field decomposition.

The difference of the presence of dynamic reconfiguration is shown in Fig. 3.

Without reconfiguration, three modules, `Module A`, `B` and `C`, are implemented on a single device. The reason is any modules cannot be added to a finished product such as ASICs. Some fine-grained architecture devices are classified in this category because their reconfiguration time is too large to do dynamic reconfiguration, which are generally used in a rapid-prototyping system. A power-gating approach [17], [25] may be used for its power reduction as shown in Fig. 3 (i). But, it can not be applied to all cases because the overhead of restarting circuits is not small. A clock-gating approach [26], [27] can become the alternative as shown in Fig. 3 (ii). It can reduce $P_{dynamic}$ but not P_{static} described in Sect. 2.2. If we would like to reduce not only $P_{dynamic}$ but also P_{static}, dynamic reconfiguration are required.

With reconfiguration, as shown in Fig. 3 (iii), we can reduce P_{static} because the basic circuit amount is cut by 25% (=3/12) in this instance. Then, some power reduction is expected at `Void` circuits if the circuit reconfiguration is scheduled from the view of a computational timeline. Moreover, a clock-gating approach can be applied to this implementation the same as Fig. 3 (ii). The difference of clock-gating approaches including our proposition is explained in the following section.

2.4 Dynamic Reconfiguration with a Clock-Gating Approach

As discussed in earlier sections, dynamic reconfiguration is an effective approach toward the reduction of the power consumption. In this section, this paper introduces a clock-gating approach for further power reduction. Figure 4 shows the difference of clock-gating approaches in dynamic reconfiguration.

The `Case A` in Fig. 4 is the most commonly used approach when a clock-gating approach is discussed. It is work on a device properly but it has not been optimized for reconfigurable devices.

The `Case B` is optimized in a temporal axis. When the amount of `Void` circuits is integrated for a computation period, it is larger than that of `Case A`. Thus, `Caes B` can achieve the larger power reduction compared to `Case A`.

The `Case A` is the most aggressive approach. It can achieve further power reduction compared with both `Case A` and `B` though the complicated design process and the knowledge of the adopted architecture are required. One thing which is important is that `Case C` requires not only a module-based spatial design but also a fine-grained temporal control.

A CGRA device, DAPDNA-2 produced by Tokyo Keiki Inc., is selected as a testbed in this paper. The next session describes the features.

2.5 Augmented Reality Applications

AR applications have struggled to satisfy both of performance and low power consumption. The Sekai Camera [28] is an AR application. The information, air tag, is displayed on the screen and users will find the information to detect the

position and direction for their target site. GPS, Wi-Fi, and electronic compasses assist processor to lead the result of air tags. It leads high accurate result. But the whole system would be complicated and its power consumption is large for long-term execution.

Scene Recognition Engine (SREngine) [29] is another AR application. It extracts feature points from a scene image taken by image sensors, and then the scene is recognized by comparing with a scene database which is prepared by prior learning. Some additional information can be provided by the recognition and users can know the place, shops, landmarks, and so on by the application. SREngine is a good technique for mobile computing because it does not need any other sensors such as GPS. However, there are two controversial points. One is SREngine that requires cloud communication. The computation is supported by SREngine servers via Internet since mobile devices have poor computational power. The other is its recognition rate which is not so high because the limited computation is not able to support all situations such as the difference of field angle and exposure.

We propose hardware direct computation as a total solution and it makes the most of the limited power resource. It has technological feasibility when the system adjusts for the situation demands with hardware reconfiguration. It has been discussed for more than a decade [30], [31] but there is few quantitative analysis of real applications on a real system. One of the contributions of this paper presents the overall assessment of an AR application on a real system.

3. DAPDNA-2 Architecture

3.1 DAPDNA-2 Architecture

Figure 5 shows the overview of DAPDNA-2 architecture. It is one type of dynamic reconfigurable architectures and the devices are delivered by Tokyo Keiki Inc [15]. It consists of the Digital Application Processor (DAP), the Distributed Network Architecture (DNA), and I/O interfaces.

DAP is a custom-made 32-bit RISC processor and any application program works as well as general processors. For instance, we can run the axLinux [32] and add web server functions on DAP. DNA includes an array which is composed of configurable processing elements, on-chip memory blocks, and so on. Its array configurations and these switch references for dynamic reconfiguration can be controlled by DAP. PCI and memory interfaces, DMA controllers and direct I/Os are also embedded on DAPDNA-2. These interfaces except for direct I/Os are connected to a high-speed bus switch.

3.2 Processing Elements and Context Memories

DAPDNA-2 adopts a coarse-grained architecture. The basic elements are classified to 32-bit ALUs, delay elements, memories, counters, and I/O interfaces. The number of processing elements (PEs) and I/Os are shown in Table 1. ALU

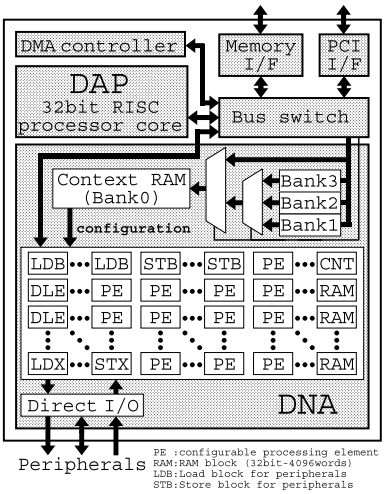

Fig. 5 DAPDNA-2 architecture.

Table 1 The number of PEs and I/Os classified to functions.

	Name	#	Function
PE	EXE	168	Arithmetic, logical & comparative operation
	DLE	136	Shift register for latency adjustment
	RAM	32	32 bit × 4,096 words per RAM element
I/O	CNT	24	I/O address generation or counter
	LDB	4	Input from asynchronous input buffer
	STB	4	Output to asynchronous output buffer
	LDX	4	Input from direct access
	STX	4	Output to direct access

functions and these connections are specified by a DNA configuration which is stored in a context RAM. The configuration stored in Bank 0 is used for the current design which works at the moment.

If multiple configurations are stored in different context memories, it facilitates the pre-fetching of a design and then supports fast reconfiguration. DAPDNA-2 has 4 multiple context memories: Bank 0, 1, 2, and 3 in Fig. 5. Bank 0 always stores a foreground configuration and the others have background configurations. A background configuration can be switched from foreground configuration in a single clock cycle, about 6 nanoseconds. If the number of configurations is larger than 4, its excess configurations will be loaded from DAP to background context memories.

4. Subspace Method

Subspace methods are a well-known approach for pattern recognition [20]. A subspace means a low-dimensional space which is generally formed by eigenvalue decomposition or principal component analysis. It provides a reasonable approximation to the set of patterns such as an image database and is very attractive because high-dimensional data analyses are unwanted, in other words, they can be represented using a small number of coefficients. For these reasons, a proper feature subspace is strongly required and

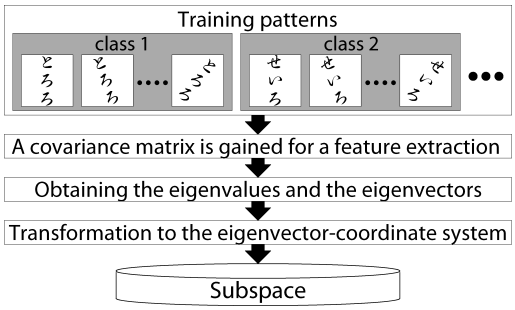

Fig. 6 Processing flow for subspace learning.

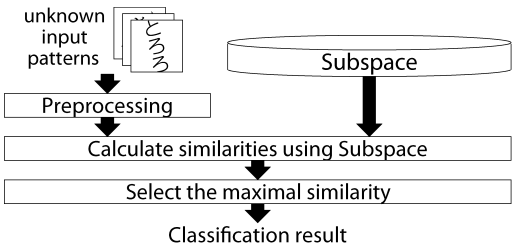

Fig. 7 Processing flow for pattern classification.

refined approaches for subspace learning are discussed.

Figure 6 shows the processing flow of subspace learning. Each image pattern which is expressed as a pixel vector $\mathbf{x}$ belongs in class ω_j. The principle component analysis algorithm calculates the covariance matrix $\mathbf{R}_j = E\{\mathbf{x}\mathbf{x}^\top | \mathbf{x} \in \omega_j\}$ and $\mathbf{R}_j$ is estimated using $\hat{\mathbf{R}}_j = n_j^{-1} \sum_{j=1}^{n_j} \mathbf{x}_j \mathbf{x}_j^\top$ where n_j is the number of dimensions for $\mathbf{x}_j$. It extracts the eigenvectors $\mathbf{u}_j$ and the corresponding eigenvalue λ_j. This CLAss-Featuring Information Compression, namely CLAFIC [33], is a basic method and widely used for the principal component analysis. The original patterns are transformed to the optimal vectors on an eigenvector-coordinate system, and then a subspace L_j is obtained.

In this pattern classification, unknown input patterns are classified by using the subspace as shown in Fig. 7. The similarities between an input vector $\mathbf{x}_j$ and the subspace are calculated by the Eq. (3).

$$\mathbf{S}_j(\mathbf{x}) = \sum_{j=1}^{d_j} (\mathbf{x}_j^\top \mathbf{s}_j)^2 \tag{3}$$

where $\mathbf{s}_j$ is the j-th eigenvector belonging to class ω_j. The $\mathbf{s}_j$ is also expressed as

$$(\hat{\mathbf{R}}_j - \lambda_j \mathbf{I})\mathbf{s}_j = \mathbf{0} \quad (j = 1, \cdots, d_j) \tag{4}$$

where λ_j is larger than λ_{j+1}. $\mathbf{x}_j$ will be classified into class ω_j which takes the maximal value of the similarity $\mathbf{S}_j(\mathbf{x}_j)$.

5. Motion Tracking by a Gradient Descent Method

The Lucas-Kanade (LK) method [21] is one of differential approaches which relies on the spatial and temporal derivatives of an image. The constraint on intensity conservation brings about a situation where the global illumination does not change but some pixels can move. In particular, when the pixel (x, y) at time t moves to $(x + \Delta x, y + \Delta y)$ at $t+\Delta t$, the following constraint equation is obtained.

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) \tag{5}$$

The intensity derivatives can be approximated by the first order of the Taylor series expansion.

$$\begin{aligned} &I(x + \Delta x, y + \Delta y, t + \Delta t) \\ &\quad = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \end{aligned} \tag{6}$$

Using the Eq. (5), the Eq. (6) becomes

$$\begin{aligned} 0 &= \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \\ &= \lim_{\Delta t \to 0} \left(\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t} \right) \\ &= I_x u + I_y v + I_t \end{aligned} \tag{7}$$

where (u, v) is the optical flow vector $\mathbf{d}$ and I_x, I_y and I_t are partial derivatives of the image intensity. In supposing that $\mathbf{d}$ is constant in a small window $(w \times w)$, the intensity values of n pixels are assumed:

$$\begin{bmatrix} I_{x_1} \\ I_{x_2} \\ I_{x_3} \\ \vdots \\ I_{x_n} \end{bmatrix} u + \begin{bmatrix} I_{y_1} \\ I_{y_2} \\ I_{y_3} \\ \vdots \\ I_{y_n} \end{bmatrix} v + \begin{bmatrix} I_{t_1} \\ I_{t_2} \\ I_{t_3} \\ \vdots \\ I_{t_n} \end{bmatrix} = 0 \tag{8}$$

Equation (9) is given by solving the Eq. (8) using the method of least squares.

$$\mathbf{A}^\top \mathbf{A}\mathbf{d} = -\mathbf{A}^\top \mathbf{b}, \tag{9}$$

then $\mathbf{A}$ and $\mathbf{b}$ can be described as

$$\mathbf{A} = \begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ I_{x_3} & I_{y_3} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} I_{t_1} \\ I_{t_2} \\ I_{t_3} \\ \vdots \\ I_{t_n} \end{bmatrix} \tag{10}$$

$\mathbf{A}^\top \mathbf{A}$ is a regular matrix and therefore $\mathbf{d}$ is given by solving the following equations.

$$\mathbf{d} = \mathbf{Z}^{-1}\mathbf{e} \tag{11}$$

where,

$$\begin{aligned} \mathbf{Z} &= \mathbf{A}^\top \mathbf{A} \\ &= \begin{bmatrix} \sum\sum_w I_x^2 & \sum\sum_w I_x I_y \\ \sum\sum_w I_x I_y & \sum\sum_w I_y^2 \end{bmatrix} \end{aligned} \tag{12}$$

$$\begin{aligned} \mathbf{e} &= \mathbf{A}^\top \mathbf{b} \\ &= -\begin{bmatrix} \sum\sum_w I_x I_t \\ \sum\sum_w I_y I_t \end{bmatrix} \end{aligned} \tag{13}$$

As presented above, it is very important that noticeable characteristic points are found in a target image. The Kanade Lucas Tomasi algorithm [34] is one of the most popular approaches for finding feature points in computer vision and the procedure is as follows:

1. Calculate **Z**'s eigenvalues in the Eq. (12)
2. Choose minimal eigenvalues
3. Detect feature points if they are larger than threshold λ

6. Implementation

6.1 System Overview

Figure 8 shows the photographs of one interface board and one DAPDNA-EP102 board.

The experiment system is composed of five components: one four-million-pixel CCD-image sensor, one controller, one head-mounted display named Vuzix iWear VR920, one interface board which is developed by this project, and one DAPDNA-EP102 board. The DAPDNA-EP102 board has one DAPDNA-2 device and one 512MBytes micro SDRAM. Figure 9 shows the system overview and the relation among these devices.

The integrated development environment named DAPDNA-FW Ⅱ is used for DAPDNA-2 design. It supports design simulation and debugging, and can send the design and program to a real device. DNA Designer can set not only PE placements and their inter connection but also PE operations and parameters through GUI. Therefore, it is possible to develop the LSI design easily without hardware description language (HDL).

The system is intended to a multilingual translational application for augmented reality. When a user wants to translate a foreign word to first language, a word which is captured as an input image is designated by the user and then automatically translated by the system. The system can provide much information including interpretive images if a rich database is prepared. For developing as a portable AR system, the system demands is too high; the system should finish all computation of a VGA frame in 33 milliseconds, the size is smaller than a cube 15 cm on a side, and the power consumption must be less than 10 watts including HMD and some peripheral devices.

Processing flow of the proposing system is shown in Fig. 10. The input image from the CCD sensor module is loaded to reconfigurable computation block in DNA. After word recognition and tracking computations, the output image is generated and sent to head-mounted display. The input-output processing is a constant flow but all circuits can not be implemented on a single device. Here, the computational block is dynamically reconfigured for the subspace method, the feature point extraction and the LK method sequentially. The configuration time is too short, less than 7ns, and therefore the configuration timing can be arranged freely.

6.2 Subspace Method Optimization

Each element of eigenvectors is expressed using 32-bit floating-point values. However, reconfigurable devices including DAPDNA are no good with multiplications, divisions and extractions of square root using broad data width such as 32-bit and 64-bit notation. To avoid this, the dictionary data is converted from floating point notations to fixed point notations. This conversion leads to save many circuits and realize highly parallel computation.

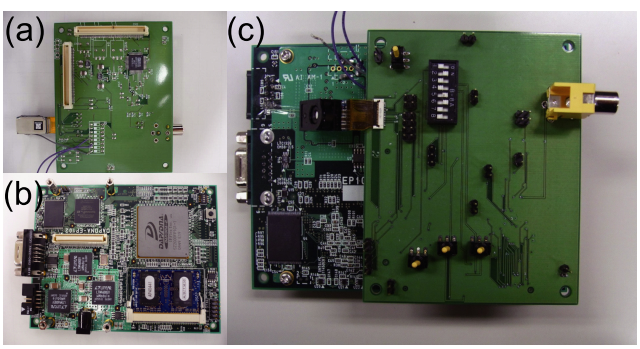

Fig. 8 (a) I/F board (the reverse side), (b) DAPDNA-EP102 board, and (c) DAPDNA-EP102 combined with the I/F board.

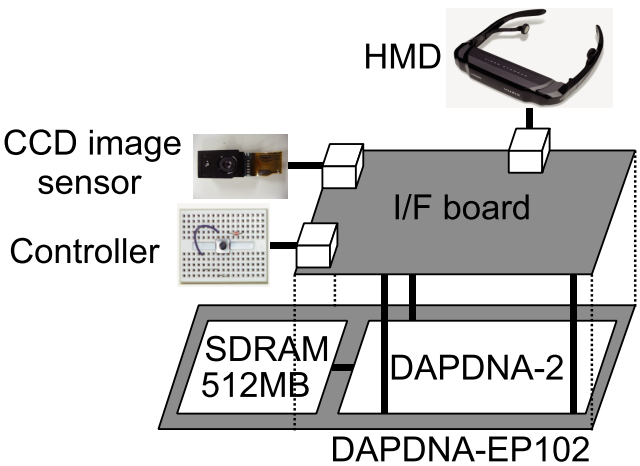

Fig. 9 System overview and device connection.

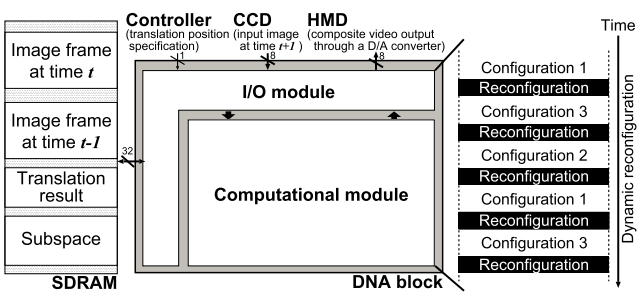

Fig. 10 Processing and reconfiguration flow.

6.3 KLT Tracker Optimization

There is a big problem how the computation circuits for eigenvalues of Z, λ_1 and λ_2, are implemented on a single DAPDNA The computation is required to solve characteristic equations of Z. That is:

$$\lambda^2 + a\lambda + b = 0 \tag{14}$$

where,

$$a = -(\sum_w I_x^2 + \sum_w I_y^2) \tag{15}$$

$$b = det\,\mathbf{Z} = det\begin{bmatrix} \sum\sum_w I_x^2 & \sum\sum_w I_x I_y \\ \sum\sum_w I_x I_y & \sum\sum_w I_y^2 \end{bmatrix} \tag{16}$$

This naive approach requires to compute extractions of square root and it has an unwanted effect on the circuit inefficiency. The proposed approach is what eigenvalues are estimated from the coefficients of a characteristic equation without computing eigenvalues. Factor in Vieta's formulas, the relationship between eigenvalue is computed:

$$\lambda_1 + \lambda_2 = -a > 0 \tag{17}$$

$$\lambda_1\lambda_2 = b > 0 \tag{18}$$

Minimum eigenvalue of good feature points to track showed plus value at least because of result of preliminary experiment. So points that minimum eigenvalue is minus value check off. Minimum eigenvalue assumed plus value ($\lambda_2 > \lambda_1 > 0$):

$$\lambda_1 + \lambda_2 > 0 \tag{19}$$

$$\lambda_1\lambda_2 > 0 \tag{20}$$

If both $\lambda_1 + \lambda_2$ and $\lambda_1\lambda_2$ are larger compared with the previously-stored values, minimum eigenvalue is larger too. Therefore the minimal eigenvalue judged from the value of coefficients.

7. Result

Through our experiment using the translation system for AR, we evaluate that the recognition and tracking algorithm which are implemented on a single DAPDNA with dynamic reconfigurations as shown in Fig. 6. In this section, we show the result of circuit implementations and the each performance evaluation. The overview of implementation results is shown in Table 2.

7.1 Development System

The outline and usage scene of our system are shown in Fig. 8 and Fig. 11 respectively. The size of the system is 13 × 10 × 14cm and the weight is about 300 g including the CCD image sensor, the head-mounted display, our interface board et al. The function of the system is tested using our AR translation application for Japanese restaurant menu which includes 26 items. The CCD image sensor takes 30 frames VGA-resolution image per a second. Then DAPDNA receives the image adequately and execute processing for recognition and tracking et al. After the processing, result image is transferred through the interface board and is displayed on the head-mounted display properly. For the menu, the system can recognize and track all items correctly.

Table 2 Implementation results: Subspace, KLT, and LK.

Processing	Subspace	KLT	LK
Number of EXE elements	41	125	150
Number of Delay elements	4	25	44
Number of RAM elements	1	26	41
Processing time (ms)	0.691	0.049	0.005
Power consumption (mW)	461	2241	3725

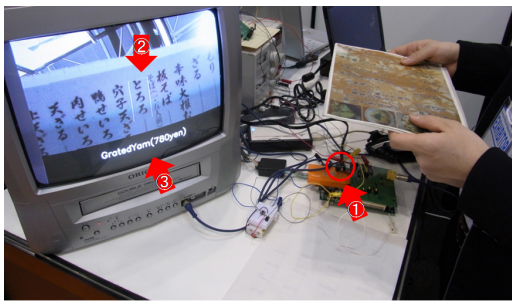

Fig. 11 ① CCD captures a whole menu, ② DNA starts to translate a word in a particular rectangle, ③ A translation result is proposed in real time.

7.2 Evaluation Experiment for Subspace Method

In experiment of recognition process, we measure precision and processing time of both implementations software and DNA. The procedure for the experiment is as follows.

1. Prepare training data and test data for each class
2. Calculate subspace of each class using corresponding training data.
3. Classify test data using the calculated subspaces.
4. Measure mean precision and processing time of each class

In our procedure, number of training data is 10,250 and number of test data is 1,000 for each class. These data are generated by applying affine transformation to template image. The processing time is between input of feature vector and output of classification result. The processing time of software implementation is measured by using Query Performance Counter function and Query Performance Frequency function which are offered by Microsoft. Specification of computer for the experiment of software implementation is Windows Vista Business 32bit OS, Intel Core 2 Duo CPU E6550 2.33GHz and 2,048MB DDR2 memory.

The counterpart program is written by C++ and OpenCV1.0 and the development environment is Microsoft Visual Studio 2008 Professional Edition 9.0.21022.8. The processing time of DAPDNA is calculated by required clock cycle until output of classification result and frequency of DNA 166 MHz. The mean precision is calculated as shown in Eq. (21).

$$mean\ precision = \frac{1}{n}\sum_{i=1}^{n}\frac{correct_i}{1000} \tag{21}$$

where n is number of class and $correct_i$ is number of test data which is classified correctly.

Result of the experiment is shown in Table 3. Here, precision of the DAPDNA implementation is estimated by fixed-point software implementation. The precision and each similarity for some test data are equivalent in DAPDNA and software implementation. In precision, fixed-point implementation fall bellow floating-point implementation by approximately 10%. In processing time, DAPDNA calculates about four times faster than the software implementation.

Table 3 Precision and processing time for subspace method.

Implementation type	Precision (%)	Time (ms)
Software (32-bit floating point)	90.25	2.62
Software (16-bit fixed point)	80.54	2.48
DAPDNA (16-bit fixed point)	80.54	0.691

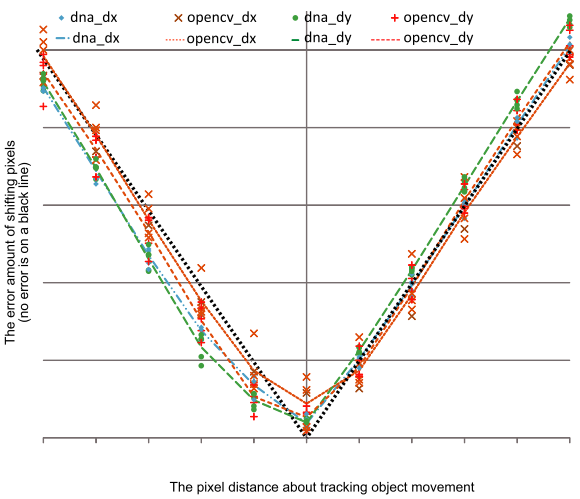

Fig. 12 Evaluation result for KLT tracker.

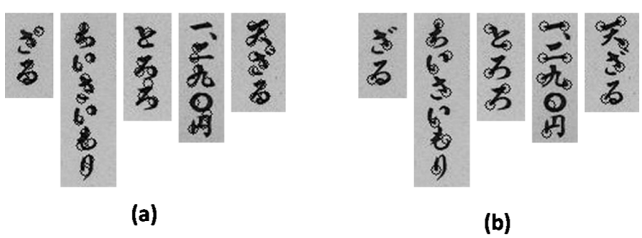

Fig. 13 Feature points (a) Our algorithm (b) OpenCV.

7.3 Evaluation Experiment for KLT Tracker

Tracking performance is evaluated by two different images experiment. One is the original image and the other is translation image. Comparative approach applied Open CV library. Result of the evaluation experiment is shown in Fig. 12. The result shows that tracking performance of our tracking algorithm is as well as OpenCV library. Results of detecting good feature points by our tracking algorithm and OpenCV library are shown in Fig. 13.

8. Evaluation Result

In the AR system, recognition process (Subspace method) and tracking process (LK + KLT method) are required for each frame and these processes should be completed within 33 *ms*.

The processing time of Subspace is 0.69 *ms* and LK + KLT is 0.05 *ms* respectively. Total processing time of the two process is 0.74 *ms* so about 98% of 1 frame time is used by these processes uneconomically.

To use the 33 *ms* more efficiently, we tested three approaches shown in Fig. 14. The `Case A` is the general approach in reconfigurable computing. The total power consumption is about 4.2 *J* and it is already smaller than general-purpose CPU.

The `Case B` is adjusting reconfiguration timing. The power consumption of Subspace is 461 *mW* and LK + KLT is 5966 *mW* so Subspace's power consumption is less than 10% of LK + KLT process's one. From this fact, assigning much time to Subspace configuration than LK + KLT configuration can reduce power consumption.

The `Case C` is introducing inactive configuration in addition to Subspace and LK + KLT configuration. Here, Subspace configuration still uses extra time. The introducing more reasonable configuration, input and output circuit only,

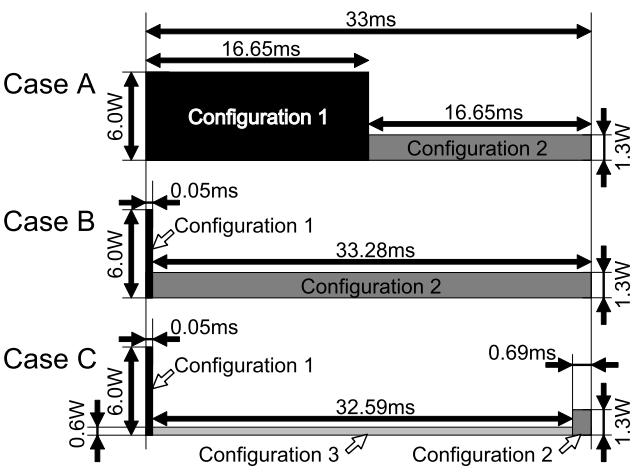

Fig. 14 Adjusting reconfiguration timing and introducing minimal configuration.

Table 4 Evaluation result of lowering power consumption.

Approah type	Average Power (J)	Power reduction ratio (%)	
		via Naive	via Optimized
Case A	4.2	0	-
Case B	1.9	△ 55.8	0
Case C	0.6	△ 85.1	△ 66.3

is considered. Assigning extra time which is used by recognition configuration to introduced configuration can reduce power consumption more than `Case A`.

Table 4 shows evaluation result for average power of each approach and how much percentage `Case C` can reduce power consumption compared with other ones. The average power of proposed method is 0.7 Joule and takes over 80% less power than naive one.

9. Conclusion

In this paper, a spatiotemporal design was discussed based on the implementation of AR application. Although fine-grained reconfigurability are desired, coarse-grained architecture was able to achieve better power-performance compared with a naive approach. It can bring SoC the significant advance of power performance because all computational circuits do not work simultaneously. Thus, reconfigurable SoCs will be one solution for not only future mobile computing but also heavily-loaded computing such as the systems in data center. However, the size of reconfigurable modules and the frequency of updating them should be considered for achieving higher power-performance. To obtain the universal rule, the trade-off among the size, the frequency, and the granularity of reconfigurable architecture must be discussed. This remains our future work.

Acknowledgments

This research was partially supported by TOKYO KEIKI Inc. and we would like to thank Mr. Watanabe for his helpful discussion.

References

[1] B. Dally, “Power, programmability, and granularity: The challenges of ExaScale computing,” Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International, p.878, May 2011.
[2] X. Fan, W.-D. Weber, and L.A. Barroso, “Power provisioning for a warehouse-sized computer,” the 34th annual International Symposium on Computer Architecture, pp.13–23, 2007.
[3] G.E. Moore, “Cramming more components onto integrated circuits,” Electronics, vol.38, no.8, pp.114–117, 1965.
[4] B. Davari, R.H. Dennard, and G.G. Shahidi, “CMOS scaling for high performance and low power-the next ten years,” Proc. IEEE, vol.83, no.4, pp.595–606, 1995.
[5] Intel turbo boost technology 2.0. http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html
[6] Amd turbo core technology. http://www.amd.com/en-us/innovations/software-technologies/turbo-core
[7] Nvidia kepler architecture. http://www.nvidia.com/object/nvidia-kepler.html
[8] H. Esmaeilzadeh, E. Blem, R.St. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” ACM SIGARCH Computer Architecture News - ISCA’11, vol.39, no.3, pp.365–376, 2011.
[9] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M.B. Taylor, “Conservation cores: Reducing the energy of mature computations,” ACM SIGARCH Computer Architecture News - ASPLOS’10, vol.38, no.1, pp.205–218, 2010.
[10] N. Goulding-Hotta, J. Sampson, Q. Zheng, V. Bhatt, S. Swanson, and M.B. Taylor, “Greendroid: An architecture for the dark silicon age,” Asia and South Pacific Design Automation Conference, pp.100–105, 2012.
[11] G. Theodoridis, D. Soudris, and S. Vassiliadis, “A survey of coarse-grain reconfigurable architectures and cad tools,” in Fine- and Coarse-Grain Reconfigurable Computing, chapter 2, Springer, 2007.
[12] A. Niyonkuru, G. Eggers, and H.C. Zeidler, “A reconfigurable processor architecture,” Field-Programmable Logic and Applications, pp.1160–1163, Sept. 2002.
[13] E. Tau, D. Chen, I. Eslick, and J. Brown, “A first generation DPGA implementation,” the Third Canadian Workshop on Field-Programmable Devices, pp.138–143, May 1995.
[14] H. Nakada, K. Oguri, N. Imlig, M. Inamori, R. Koniski, H. Ito, K. Nagami, and T. Shiozawa, “Plastic cell architecture: A dynamically reconfigurable hardware-based computer,” Parallel and Distributed Processing, pp.679–687, April 1999.
[15] T. SugiwaraIde, K. Ide, and T. Sato, “Dynamically reconfigurable processor implemented with IPFlex’s DAPDNA technology,” IEICE Trans. Inf. & Syst., vol.E87-D, no.8, pp.1997–2003, Aug. 2004.
[16] V. Lari, S. Muddasani, S. Boppu, F. Hannig, M. Schmid, and J. Teich, “Hierarchical power management for adaptive tightly-coupled processor arrays,” ACM Transactions on Design Automation of Electronic Systems, vol.18, no.1, pp.1–25, 2013.
[17] D. Kissler, D. Gran, Z. Salcic, F. Hannig, and J. Teich, “Scalable many-domain power gating in coarse-grained reconfigurable processor arrays,” IEEE Embedded Systems Letters, vol.3, no.2, pp.58–61, 2011.
[18] K. Kinoshita, T. Okamura, D. Takano, T. Yao, and Y. Yamaguchi, “An augmented reality system with a coarse-grained reconfigurable device,” ACM SIGARCH Computer Architecture News — HEART2012, vol.40, no.5, pp.16–21, 2012.
[19] K. Kinoshita, T. Okamura, D. Takano, T. Yao, and Y. Yamaguchi, “Spatiotemporal modular arrangement for energy efficient reconfigurable socs,” the IEEE 7th International Symposium on Embedded Multicore System on Chips, pp.97–100, 2013.
[20] E. Oja, Subspace Methods of Pattern Recognition, Research Studies Press, 1983.
[21] B.D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” International Joint Conference on Artificial Intelligence, vol.2, pp.674–679, Aug. 1981.
[22] K. Tatas, K. Siozios, and D. Soudris, “A survey of existing fine-grain reconfigurable architectures and CAD tools,” in Fine- and Coarse-Grain Reconfigurable Computing, chapter 1, Springer, 2007.
[23] Altera Corp. Stratix ii and virtex-4 power comparison. http://www.altera.com/devices/fpga/stratix-fpgas/stratix-ii/stratix-ii/features/st2-90nmpower.html, July 2008.
[24] Altera Corp. Stratix ii 90-nm silicon power optimization. http://www.altera.com/devices/fpga/stratix-fpgas/stratix-ii/stratix-ii/features/st2-90nmpower.html, April 2008.
[25] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, “Microarchitectural techniques for power gating of execution units,” Low power electronics and design, pp.32–37, Aug. 2004.
[26] F. Theeuwen and E. Seelen, “Power reduction through clock gating by symbolic manipulation,” Logic and Architecture Synthesis,

pp.184–191, 1996.
[27] S. Huda, M. Mallick, and J.H. Anderson, "Clock gating architectures for FPGA power reduction," Field Programmable Logic and Applications, pp.112–118, Aug. 2009.
[28] Sekai camera. http://sekaicamera.com/
[29] Scene recognition engine. http://www.srengine.com/
[30] W. Luk, T.K. Lee, J.R. Rice, N. Shirazi, and P.Y.K. Cheung, "Reconfigurable computing for augmented reality," Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp.136–145, 1999.
[31] J. Toledo, J.J. Martínez, J. Garrigós, R. Toledo-Moreo, and J.M. Ferrández, "Design of embedded augmented reality systems," in Augmented Reality, chapter 3, InTech, 2010.
[32] axlinux. http://www.axlinux.com/english/index.html
[33] S. Watanabe, P.F. Lambert, C.A. Kulikowski, J.L. Buxton, and R. Walker, "Evaluation and selection of variables in pattern recognition," Computer and Information Sciences, vol.2, pp.91–122, 1967.
[34] J. Shi and C. Tomasi, "Good features to track," IEEE Conference on Computer Vision and Pattern Recognition, pp.593–600, June 1994.

Kei Kinoshita received his ME degree from the University of Tsukuba, Japan, in 2010. He currently works in a internet service provider.

Yoshiki Yamaguchi received his Ph.D. degree from University of Tsukuba in 2003. He was selected as a JSPS Research Fellow from 2000 to 2003. In 2003, he joined RIKEN, Yokohama, Japan, as a research scientist. Since 2005, he has been an assistant professor of the faculty of Engineering, Information and Systems, University of Tsukuba. From 2005 to 2008, he had been a Visiting Researcher at RIKEN. From 2010 to 2011, he had been a Visiting Scientist at Imperial College London, UK. Since 2011, he has been a Collaborative Fellow at Center for Computational Sciences, University of Tsukuba. His research interests include reconfigurable, parallel, and high-performance computing. He is a member of IEICE, IPSJ, ACM, and IEEE.

Daisuke Takano received the BE and ME degrees from the University of Tsukuba, Japan, in 2008 and 2010, respectively. He currently belongs to a certain automotive supplier.

Tomoyuki Okamura received his ME degree from the University of Tsukuba, Japan, in 2010. He currently works in a certain automotive components manufacturer.

Tetsuhiko Yao received the M.Eng degrees in computer science from the University of Tsukuba in 2010. He has been involved in parallelization of image processing.