An Adaptive Pattern Recognition Hardware with On-chip Shift Register-based Partial Reconfiguration

Hiroyuki Kawai, Yoshiki Yamaguchi, Moritoshi Yasunaga

Graduate school of
Systems and Information Engineering
University of Tsukuba
1-1-1 Ten-ou-dai, Tsukuba, Ibaraki, Japan

kawai@islab.cs.tsukuba.ac.jp

Kyrre Glette, Jim Torresen

Department of Informatics
University of Oslo
P.O.Box 1080 Blindern, N-0316 Oslo, Norway

{kyrrehg, jimtoer}@ifi.uio.no

Abstract

A pattern recognition system that can process a large amount of image data at high speed is required in many fields. In this paper, we propose an on-chip pattern recognition system that utilizes the reconfigurability of the FPGA. The features of the system are not only very high recognition speed but also an adaptive function. For example, when objects to be detected change appearance, recognition parameters must be changed to retain the recognition accuracy. The system can automatically adjust by executing on-chip partial reconfiguration. The system runs at 25MHz and can return a recognition result in one clock cycle, 40*ns. To update the system, all processes needed for searching for the best recognition parameters, generating configuration data and reconfiguring the system are carried out within* 30*s.*

1. Introduction

Image recognition is a computationally demanding task where implementation in hardware is often required. Flexibility to incorporate new pattern data could then often be limited. To provide increased flexibility we have earlier proposed a method called Direct Data Implementation (DDI) in [1]. In this method, pattern data, e.g. images, are directly transformed into logic circuits and implemented in reconfigurable hardware. Since the implemented circuits are only combinational circuits, they are processed simultaneously and the structure of the system is parallel. Thus, these features make it possible to realize a system with a very high speed (described in Section 2.2 in detail). In addition, pattern data varies in each application, so that ASICs are not suitable for this method. We use FPGAs instead to take advantage of the FPGA's reconfigurability. In fact, each application can be implemented one at a time on the same FPGA chip and when a part of the pattern data is changed, it can be reconfigured by generating a new bitstream. Therefore, DDI is a method that utilizes the features of the FPGA.

In order to develop a practical system using DDI, a pattern recognition method [1, 2], namely Kernel-Based (K-B) method (see Section 2.1) was implemented in hardware with DDI to construct a pattern recognition hardware system. As a result, the system achieved high speed, acceptable small size, and high recognition accuracy. However, when a part of the pattern data is changed, the constructed system must be stopped for reconfiguration. In addition, a new FPGA bitstream must be generated with an external PC and it takes long time for such a task. Then it is difficult to update the system immediately according to new pattern data. But such a system must be promptly updated for practical use. E.g. when a face image is registered in an authentication system, it is desirable that one can use it immediately after registration.

To overcome the difficulties, we improve the offline DDI system in this paper by constructing an on-chip system that applies DDI and the K-B method. This system can dynamically reconfigure a part of the implemented circuits. In fact, when pattern data is changed, new configuration data is generated in the system itself and partial reconfiguration could automatically be undertaken. Furthermore, it is possible to generate new configuration data and update the system immediately since configuration data for the system can be produced without logic synthesis. The reconfiguration is carried out by shift operations. More specifically, LUTs in an FPGAs are instantiated as shift registers and configuration data is sent to them from a PowerPC core embedded in the FPGA. As a result, on-chip and very fine-grained, LUT-level, partial reconfiguration can be achieved.

Several on-chip reconfiguration systems have been proposed so far. However, they can not solve our aim. A typical partial reconfiguration system supported by Xilinx exists [8]. In the system, several partial bitstreams are stored in a memory device such as CompactFlash in an FPGA board. Partial reconfiguration is then executed through ICAP (Internal Configuration Access Port) of an FPGA. However, in this system, partial bitstreams must be pre-synthesized by a PC. Thus, it is impossible to generate new configuration data when the system is in operation and it can therefore not incorporate new pattern data. In our system, configuration data can be dynamically generated in operation with new pattern data and we do not use ICAP for partial reconfiguration.
An on-chip reconfiguration system by shift operations has been proposed by Tufte et al [4, 5]. They proposed a novel structure named *Sblock* that consists of 2 LUTs (Look Up Tables), namely LUT F and LUT G in a slice. That is a somewhat complex structure and not needed in our system. In our work, shift registers (SRs) are just used with cascade connections. Thus, a simpler system can be built. In addition, we apply the constructed system to a practical application, face image recognition.
On the other hand, Glette et al [6] have developed an on-chip classification system. However, our classification algorithm and a reconfiguration approach are different from theirs. A comparison of the two systems is included in Section 5.2.

In the next section, the pattern recognition approach to be implemented on an FPGA is introduced. Section 3 describes the system that we have developed and the dynamic Partial Reconfiguration approach for the system. In Section 4 and 5, experiments and effectiveness of the system are described, respectively. Finally, we conclude the paper in Section 6.

2. Direct Data Implementation for Pattern Recognition

Before describing DDI, a pattern recognition approach, the K-B method, that we adopt in this work is explained including how it is extended for hardware implementation. Next, the main approach of this work, DDI, is explained and then a pattern recognition circuit combining DDI with the K-B method is outlined.

2. 1. Extended Kernel-Based Method

In general pattern recognition processes with the K-B method , the discrimination function $D_i(\cdot)$ is made as a function of a pattern X that is an n-dimensional vector, $X = (x_1, x_2, ..., x_k, ..., x_n)$ for each category C_i. X is then classified in C_i with the following rule:

$$if\ D_i(X) > D_j(X)\ \ then\ \ X \in C_i\ \ (\forall j \neq i) \tag{1}$$

If the K-B method is implemented in VLSIs, its circuit becomes very complex because the Gauss function is often used for it. Thus, the K-B approach was then expanded in order to avoid the Gauss function and implemented on an FPGA with only simple circuits in [1]. Here, $D_i(X)$ is as follows:

$$D_i(X) = \sum_{j=1}^{N_i} K^*(X - S_j^i) \tag{2}$$

$$K^*(x) = \begin{cases} 1\ :\ |x_k| \leq d \\ \qquad\quad \forall k \in \{1, 2, ..., n\} \\ 0\ :\ otherwise \end{cases} \tag{3}$$

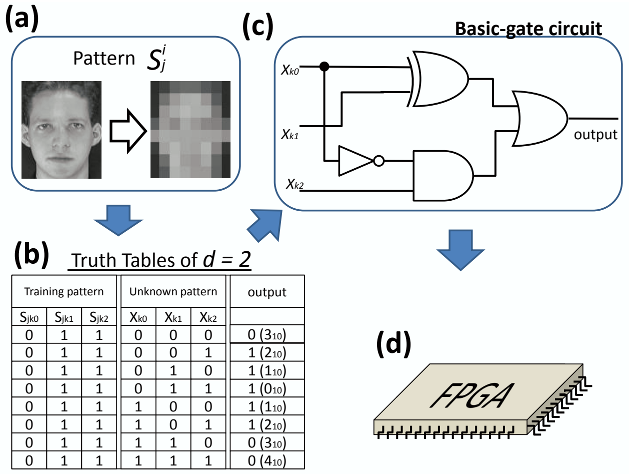
where K^* is a function that returns 1 if the pattern X is within the hypercube that is located at S_j^i of which each edge is d. S_j^i is the j-th training pattern of category i and N_i is the number of training patterns that belong to C_i.

Eq.3 makes it easy and simple to implement the method in hardware because it can be realized by only combinational logic circuits as described in the next section.

2. 2. Direct Data Implementation (DDI)

In this section, we describe DDI approach [1] that implements pattern data on FPGAs. In Eq.3, both a training pattern S_j^i and the d value are constant numbers, so they can be represented as a truth table that returns "1" or "0" according to an input pattern X. Figure 1 shows the procedure of DDI. Each step is explained below.

1. In Fig.1(a), the pattern database stores pattern data with information about their categories (e.g. the face image, where j-th data belongs to category i). In the example, the face image is translated into a downscaled image, 8 x 8 pixels, where each pixel is quantized by 3-bit precision (the right image in (a)). Then, one pixel is selected as an example and assuming that its value is "011" as shown in Fig. 1(b). The reason why 3-bit precision is adopted here is that it is easy to explain as an example. In the experiment of this paper (Section 4.), 4-bit precision is adopted.



Training pattern			Unknown pattern			output
S_{jk0}	S_{jk1}	S_{jk2}	X_{k0}	X_{k1}	X_{k2}	
0	1	1	0	0	0	0 (3_{10})
0	1	1	0	0	1	1 (2_{10})
0	1	1	0	1	0	1 (1_{10})
0	1	1	0	1	1	1 (0_{10})
0	1	1	1	0	0	1 (1_{10})
0	1	1	1	0	1	1 (2_{10})
0	1	1	1	1	0	0 (3_{10})
0	1	1	1	1	1	0 (4_{10})

Figure 1: Overview of Direct Data Implementation.

2. In Fig.1(b), the selected pixel is called *training pattern*. We use it to make a truth table. As an example, we set the d value to 2. In the truth-table, $|trainingpattern(S_{jk}^i) - Unknownpattern(X_k)|$ is calculated. The parenthetic decimal values in the *output* column are $|S_{jk}^i - X_k|$. If the difference is 2 or less, the corresponding output is 1. Otherwise, it becomes 0.

3. In Fig.1(c), a circuit is created according to the generated truth table. We name the circuit *basic-gate circuit*. In fact, one truth table is translated to one basic-gate circuit. This circuit represents the truth table of (b).

4. In Fig.1(d), the generated circuit is implemented on an FPGA. As a result, DDI is achieved. For each category, a number of such circuits are included as described in Section 2.3.

As we show, basic-gate circuits can be generated from training pattern and the d value. Thus, deciding the most appropriate d value is very important to obtain the highest recognition performance. In [1], the best d value is found by a full search on a PC. More specifically, the range of the d value is from 0 to $2^{bitprecision} - 1$, i.e. $2^3 - 1 = 7$ in the example in Fig.1. To find the optimal value of d, it is incremented by one from 0 and the accuracy is calculated with training pattern data for each d value. Finally, the d value that obtains the best accuracy is the one used in the final recognition circuit.

2.3. Kernel-function Circuit

Figure 2 shows *kernel-function circuit* that represents a whole training pattern S_j^i in Fig.1 meanwhile

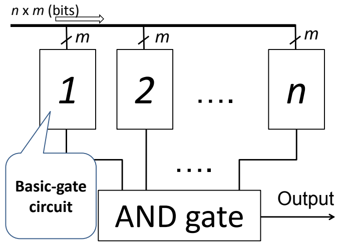


Figure 2: A kernel-function circuit that represents one training pattern.

a basic-gate circuit regards only one pixel in the image. The n value in Fig.2 represents the number of basic-gate circuits. Each basic-gate circuit is connected to an AND-gate and its output represents Eq.3. When the output is '1', it involves the whole training patterns are matched for all the pixels. Experiments on reducing the number of pixels input have not yet improved the recognition performance. If the pattern(S_j^i) is 64 pixels (8x8) and 3-bit precision, n is 64 and m is 3 in Fig.2.

2.4. The Pattern Recognition Circuit

Finally, we assemble the pattern recognition circuit (Fig.3). All kernel-function circuits are clustered in a category by counting outputs with "1" — called ones counter. The output value of each counter represents Eq.2. Then, by adding a Max Detector that selects the highest value from the counters, it is discriminable to which category the input pattern belongs. The size of the pattern recognition circuit depends on the number of training data. If the number of face images as training data is 200, 200 kernel function circuits are required. In addition, we must include counters equal to the number of categories.

As stated, almost all components such as basic-gate circuits and kernel-function circuits in the DDI method depend on pattern data. That is the reason why we call this approach DDI. In addition, we do not use any sequential circuit in the circuit and all kernel-function circuits can process input data in parallel. Thus, a very high speed pattern recognition circuit can be achieved.

3. On-chip DDI system

3.1. Overview of the System

Figure 4 shows the complete system that we have developed in this work. It is implemented on a Virtex-II Pro FPGA (XC2VP30, XUP board).

The system mainly consists of 5 units and they are connected by the on-chip peripheral bus (OPB Bus). ARA (Adaptive Recognition Area) is the most

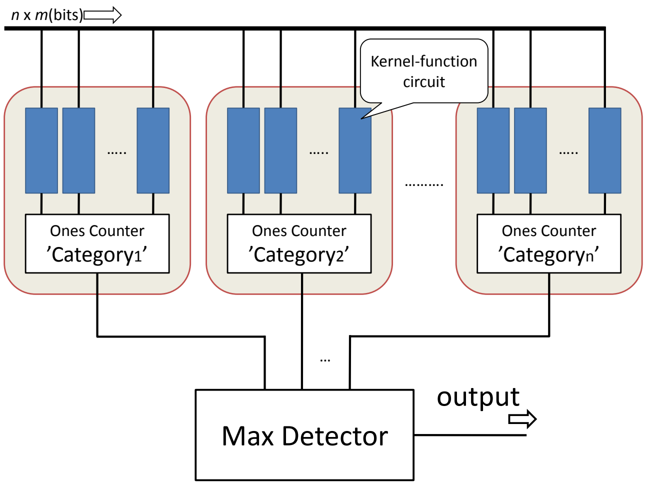


Figure 3: The pattern recognition circuit.

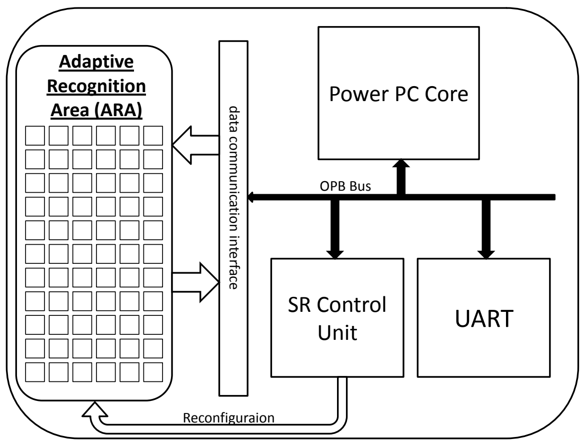
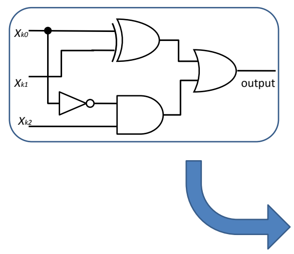


Figure 4: The overall structure of the on-chip system.

important part of the system. The recognition circuit shown in Fig.3 constitutes the ARA. It is modified in the on-chip reconfiguration system as described in Section 3. 2. Configuration data for the ARA is generated by the PowerPC Core (PPC) and sent to the ARA through the SR control unit. This unit has some FIFO buffers and sends data one by one bit. The data communication interface sends test data to the ARA and receives a recognition result back from the ARA. UART (Universal Asynchronous Receiver/Transmitter) is an IP core for debugging of the system and displays the result from the ARA on an external PC.



INPUT				out
*	X_{k0}	X_{k1}	X_{k2}	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Figure 5: A LUT that represents a basic-gate circuit.

3. 2. Improvements of the On-chip DDI System

A part of the DDI architecture must be changed to enable it to be implemented in the on-chip system. We modify the basic-gate circuits in order to allow for partial reconfiguration by shift operations. The basic-gate circuit explained in Fig.1 is mapped onto a 4-input LUT as shown in Fig.5. The MSB of the LUT becomes don't care since the basic-gate circuit is only 3-bit. As we mentioned in Fig.1, basic-gate circuits are generated from truth-tables. Therefore they can be mapped onto LUTs without any overhead. Even if a basic-gate circuit has more than 4-bit input, it can be represented by using several LUTs and carry-chains, MUXCY[9], to connect each LUT. In fact, we did not specify that a basic-gate circuit was mapped onto a LUT in the conventional DDI system. Thus, that system did not have flexibility for partial reconfiguration.

Figure 6 shows the mechanism of partial reconfiguration of the system briefly. After mapping the basic-gate circuits onto LUTs, each LUT is instantiated as shift registers that also will work as a normal LUT and basic-gate circuit specifications are shifted into it. Then a shift-input line is connected to a register in the PPC and data for changing contents of the SR is sent through it. The data is called *configuration data* in this work. By changing contents of the SR, any basic-gate circuit can be represented. Thus, dynamic and on-chip partial reconfiguration is realized by this mechanism. We call such LUTs that works as basic-gate circuits *Adaptive basic-gate circuit* (ABG circuit) to differentiate from the conventional structure.

3. 3. Shift Registers for the System

Figure 7 shows two kinds of SRs used in the system. In many Xilinx FPGAs, their LUTs can be instantiated as SRL16E or SRLC16E [9]. The difference

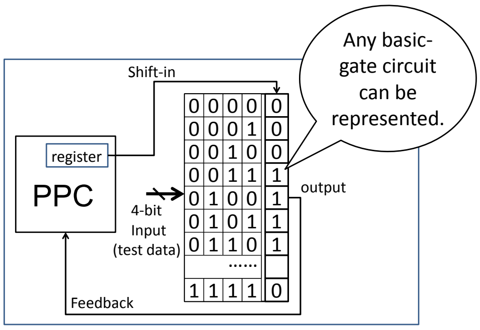


Figure 6: Shift operation in adaptive basic-gate circuit (ABG circuit) with PPC.

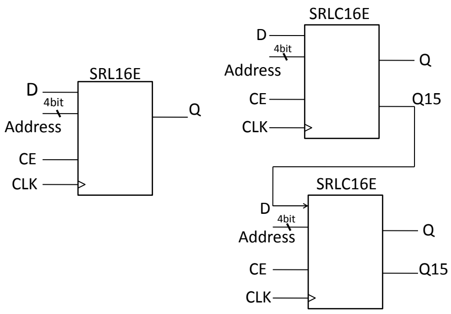


Figure 7: Shift registers used in the system

between the SRs is Q15 output of SRLC16E. The last bit of the SR is shifted out on the output. If it is connected to the D input of the next SR, shown to the right in Fig.7, a 32-bit SR can be built. This feature is very useful in our system. If the D input of each SR is connected to PPC's register, a large number of registers are required. In fact, we can save many registers of a PPC since many SRs are instantiated as SRLC16E and they are cascaded in our system. A disadvantage of this connection is that more time is required for reconfiguration. For example, if three SRs are connected by this way, it takes three times as long configuration time compared with one SR. Furthermore, the three SRs are unavailable during reconfiguration. In contrast, when we prepare three SRL16Es and they are connected independently to PPC's registers, only one SR is unavailable for reconfiguration and the others work correctly. Therefore, the performance of an online system depends on how many SRL16Es and SRLC16Es we instantiate in the system.

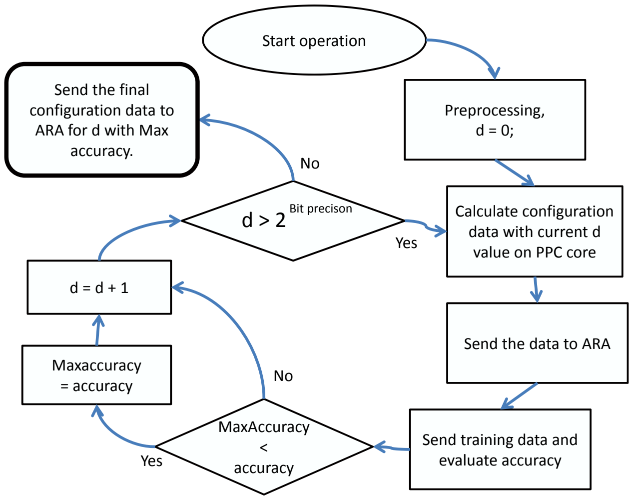


Figure 8: Flowchart of the on-chip adaptive operation.

3.4. On-chip Adaptation

When training pattern data is to be changed during operation, ABG circuits must be modified immediately to retain the recognition performance of the system. In the developed system, it can be achieved automatically without any external tools such as a PC. Here, we show how the on-chip system carries out such an operation. In Fig.8, a flowchart shows the procedure.

First, initialization of the system is done. Here, the d value is set to '0'. Next, configuration data for the ARA is calculated on PPC and the data is sent to the ARA. Then, training pattern data is sent for evaluation and the recognition accuracy of the d value equal to 0 is worked out. After the evaluation, the accuracy is stored as the current best together with the d value. The d value is then incremented by one. If the incremented value is less than $2^{bitprecision}$, the above operation is repeated and the accuracy is calculated for each value. Finally, when finishing calculating all d values, the configuration data resulting in the best accuracy is recalculated and it is sent to the ARA as the configuration data to be used for recognition.
Thus, configuration data of the highest accuracy is produced in the on-chip system automatically when training pattern data is changed and the system can retain the best recognition performance anytime. This operation is called *adaptive operation*.

4. Experiments

4.1. Preprocessing

We have applied the on-chip system to face image recognition. For this task, we used the benchmark database provided by Olivetti Research Laboratory. The database has 400 face images belonging to 40 people with 10 images each ,that is, there are 40 categories in this application. Each original image is 10,304 (92 x 112) pixels, and then they are preprocessed to downscaled images that are 64 (8 x 8) pixels and 4-bit precision. This is a common method applied in related work [1].

We use 360 images (9 images per category) for the adaptive operation. They are selected randomly that is the same approach as [1]. Since a pixel in an image is quantized by 4-bit precision, one LUT is used for a pixel in the ABG circuit and all bits in a LUT are utilized unlike in Fig.5. In the system, a kernel-function circuit has 64 (= 1 LUT x 64 pixels) LUTs and an AND-gate. Finally, 23040 (64 x 9 images x 40 categories) LUTs are used for all ABG circuits, of which SRLC16E is 23000 and SRL16E is 40. This implies that 576 LUT (575 SRL16E and one SRLC16E) that represents one category are cascaded and partial reconfiguration is executed for each category.
The remaining 40 images called *test data* are used for evaluation. They are stored in FIFO buffers in the data communication interface in Fig.4 and input to the ARA after the adaptive operation is finished.

4. 2. Device Utilization and Clock Speed

As stated above, the 360 images, 40 categories, are implemented on the XC2VP30 FPGA. It has 27,392 LUTs and is a medium-size FPGA in Virtex-II Pro series. However, when all of the images are implemented on it, 32,919 LUTs, that is 120% of the FPGA, are used. Hence the system can be implemented in only two XC2VP30 FPGAs. In addition, just one FPGA is enough when an XC2VP40 FPGA that has 38,784 LUTs is used for the implementation. In this work, we divide them into 2 blocks, that is 20 categories are implemented at a time in the FPGA. Table 1 includes device utilization when 20 categories, 180 images, are implemented. Then, 18,018 LUTs are used. Table 2 shows the usage of LUTs. SRs account for more than 60% while logic circuits account for 26%. The components for the on-chip system shown in Fig.4, such as SR control unit, UART and data communication interface, use around 4,000 LUTs. Thus, making the on-chip DDI system does not have a significant disadvantage.

According to the synthesis report from EDK, a maximum clock frequency of the system is 25.272MHz. The system behaves as very high speed pattern recognition hardware although the clock speed is not fast. Because it does not require many clock cycles for execution. Just one clock, $40ns$, is required to obtain the result from the Max Detector. Therefore, the low clock frequency does not matter in the system. In fact, the PPC runs at 100MHz while the rest of the system such as the ARA runs at 25MHz.

Resource	Used	Available	Usage
Slices	9,999	13,696	73 %
4 input LUTs	18,018	27,392	65 %

Table 1: Device utilization of the system when 20 categories are implemented on the XC2VP30 .

Function	Used	Usage
logic	4,691	26 %
route-thru	1,546	8.6 %
dual-port RAMs	206	1.2 %
shift registers	11,575	64.2 %

Table 2: Usage of the 18,018 LUTs in Table 1.

4. 3. Recognition Accuracy and Configuration Time

Figure 9 shows the recognition accuracy with training data when the d value is changed from 0 to 15. When $d = 3$, the maximum accuracy, 94.6%, can be obtained. Then 89.25% is obtained with test data as the black dot in Fig.9 shows. When the d value is more than 9, the accuracy becomes almost 0%. The reason is that almost all the content of each LUT becomes "1" when the d value is large. Then, the ABG circuits output almost only "1" every time and it is hard to recognize correctly.

Next, we present effectiveness of the adaptive system. It often occurs that light intensity of face images suddenly changes due to the sun, lamps, etc. Typically, recognition accuracy then deteriorates. In this case, alternative images as training data which are under the same condition must be prepared and reconfigure the recognition circuit according to the images in order to keep the recognition performance. Figure 10 shows an experimental result when the light intensity of face images is changed. First, the light intensity of test data is much changed resulting in an accuracy of 49.25%. Next, light intensity for training images are then changed with the same amount as the test images and applied to update the corresponding ABG circuits. When the number of updated images in a category is 1, the accuracy is 63.75%. When it is 9,

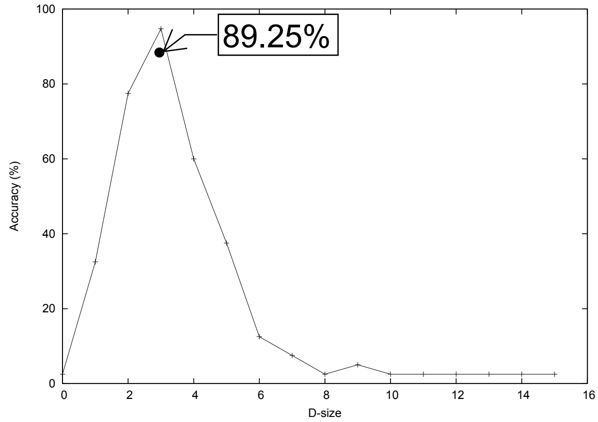


Figure 9: The accuracy of each d value.

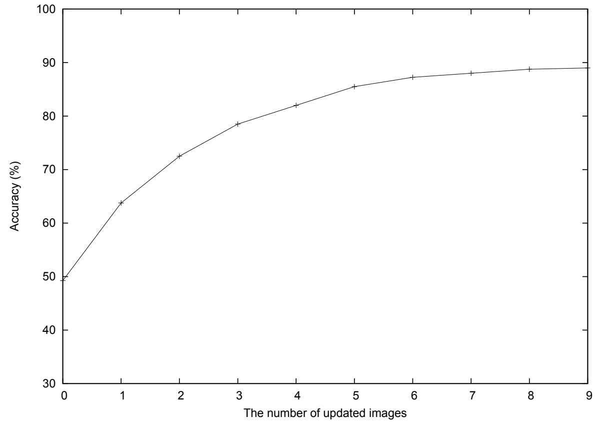


Figure 10: The accuracy when light intensity of images is changed.

accuracy is 89.00%. From the result, it is indispensable to adjust training images immediately to retain the accuracy. If the conventional DDI system in [1] is used, a new bitstream is generated by a PC each time and the system stops during reconfiguration. This is a time-consuming task. Table 3 shows how much time it takes to update the system. In the on-chip DDI system, $13.83s$ is required for all processes, namely finding the best d value, generating configuration data, and reconfiguring the system by partial reconfiguration. In addition, it is not necessary to reconfigure all categories when training data of only one category is changed. In this case, it takes only $1.14s$ to update one category while the other categories work correctly. Even if 40 categories are updated, it takes $27.18s$, within $30s$. However, it takes $1,409s$ to generate a new bitstream with a Xilinx tool on a PC (Pentium4, 2.6GHz) when the conventional offline DDI system is used. It is much slower than the on-chip system. From the result, our system can update itself approximately 50 times faster than the conventional DDI system.

the on-chip DDI, 20 categories	13.83 s
the on-chip DDI, one category	1.14 s
the on-chip DDI, 40 categories	27.18 s
the conventional DDI with a PC	1,409 s

Table 3: Update time for the on-chip system and the conventional PC-based system.

5. Discussion

5. 1. Performance of the Online System

The developed system uses a large number of SRs as mentioned in Section 4.1. During reconfiguration, a part of SRs does not output a correct value. However, it is possible to improve the performance. We suggest some approaches below to improve it from the viewpoint of both hardware and software.

In the hardware architecture, 576 SRs are currently used as one partial reconfiguration unit. When SRLC16Es are used, it is possible to construct a more fine-grained unit as mentioned in Section 3. 3. In this case, configuration time per unit can be shortened and then the number of available SRs in the system increases although the total configuration time does not change.

Or the software programs, the flow shown in Fig.8 can be optimized. In fact, when the d value is more than nine in Fig.9 the accuracy is almost 0%. Thus, the system does not need to calculate the accuracy then. In this case, calculation time becomes approximately halved and the performance of the online system is improved. In addition, if the best d value is already known, it is not necessary to repeat the operation in Fig.8. In other words, reconfiguration for each category is executed only once. In this case, to reconfigure 20 categories once, only $1.10s$ is required and to reconfigure one category once, it takes only $0.11s$ for update. From the result, required time for updating the system can be shortened to improve the performance of the online system.

5. 2. Comparison with the Other On-chip System

As mentioned in Section 1., Glette et al [6] have developed an on-chip pattern recognition system with VRC (Virtual Reconfigurable Circuit) [7] and a genetic algorithm. A comparison of that system with ours:

1. **Recognition Speed**
In our system, the pattern recognition process can be executed by just one clock cycle and only $40ns$ is required for the process. In contrast, Glette's system requires approximately $1\mu s$ for performing recognition. Thus, our system does recognition faster.
2. **Reconfiguration speed and time for updating the system**
In Glette's system, it realizes very fast reconfiguration because of using VRC. In the system, reconfiguration is through registers controlling by MUXes. Then it requires less clocks for reconfiguration than our system. However, the system adopts GA for calculation and to generate new configuration data. It takes $140s$ to finish updating the system. In our system, reconfiguration is achieved by shift operations. Thus, the performance of an online system having uninterrupted operation during reconfiguration is inferior to Glette's system. But only $27.18s$ is required to achieve all tasks because the search for the best d value is simpler than GA. Totally, our system is faster updated although it is not easy to compare them equally.
3. **Recognition Accuracy**
In our system, the best accuracy is 89.25% while Glette's one is 96.25%. We confirmed that 92.0% can be obtained when one pixel is quantized as 8-bit precision instead of 4-bit although our system is still inferior to Glette's one.

6. Conclusions

We have succeeded to develop an adaptive pattern recognition system with on-chip partial reconfiguration. In the system, configuration data for an FPGA can be generated in the system itself and reconfiguration is executed with shift registers instantiated from LUTs. While one category in the system is reconfigured, the other categories can continue to operate. Thus, the system does not need to suspend during the adaptive operation and the operation finishes in $27.18s$. In addition, it is possible to improve the configuration performance by optimizing the hardware architecture and software programs executed by a PPC in the FPGA. The system inherits one of the most important features of the DDI method, high recognition speed. The recognition process can be executed in $40ns$ in the system. On the other hands, the recognition accuracy of the system is 89.25%. In the ongoing work, we improve the accuracy, explore other applications for the system and develop more optimized architecture.

Acknowledgments

This research is partly funded by the Research Council of Norway through Cultural Agreement between Norway and Japan, Norwegian Government scholarship 2007/2008.

References

[1] Yasunaga, M., Takami, T., Yoshihara, I., "Image Recognition Hardware using FPGAs for Nano-Second Range Recognition Speed," The trans. of the institute of electronics, information and commuication engineers, in Japanese, Vol.J84D-II No.10, pp.2280-2292, October 2001.

[2] Yasunaga, M., Nakamura, T., Yoshihara, I., Kim J., "The Kernel-based Pattern Recognition System Designed by Genetic Algorithms," IEICE Transaction on Information and Systems, Vol.E84-D, No.11, pp.1528-1539, November 2001.

[3] Yasunaga M., Yoshihara I., "Development of a Pattern Recognition Evolvable Hardware System: Its Application to the Sonar Spectrum Recogniton," The trans. of the institute of electronics, information and commuication engineers, in Japanese, Vol.J86D-I No.1, pp.1-13, January 2003.

[4] Tufte, G., Haddow, P., "Biologically-inspired: A rule-based selfreconfiguration of a virtex chip," In 4th International Conference on Computational Science 2004 (ICCS 2004), Lecture Notes in Computer Science, pp. 1249-1256. Springer, 2004.

[5] Haddow, P., Tufte, G., "Bridging the genotype-phenotype mapping for digital FPGAs," In The 3rd NASA/DoDWorkshop on Evolvable Hardware, pages 109-115, IEEE Computer Society 2001

[6] Glette, K., Torresen, J., Yasunaga, M., "Online Evolution for a High-Speed Image Recognition System implemented On a Virtex-II Pro FPGA," In Proc. of the NASA /ESA Conference on Adaptive Hardware and Systems(AHS-2007), pp. 463-470, IEEE Computer Society, 2007.

[7] Sekanina, L., Ruzuicka, R., "Design of the Special fast recofngirurable chip using common FPGAs," In Proc. of Design and Diagnostics of Electronic Circuits and Systems, IEEE DDESC'2000, pp.161-168, 2000.

[8] Papadimitriou, K., Anyfantis, A., Dollas, A., "Methodology and Experimental Setup for the Determination of System-Level Dynamic Reconfiguration Overhead" In Proc of Field-Programmable Custom Computing Machines, pp. 335-336, 2007.

[9] XILINX, "Virtex-II Pro and Virtex-II Pro X FPGA User guide," `http://www.xilinx.com/`

[10] XILINX, "Xapp290," `http://www.xilinx.com`