Evaluation of the Hardware Specialization Techniques for Vibration Control Applications

Yasuaki Tezuka*
* Dept. Knowledge-based Information Engineering
Toyohashi University of Technology
Tempaku, Toyohashi 441-8580, Japan

Shuichi Ichikawa†
† Dept. Electrical and Electronic Information Engineering
Toyohashi University of Technology
Tempaku, Toyohashi 441-8580, Japan
E-mail: ichikawa@ieee.org

Yoshiyuki Noda‡
‡ Dept. Mechanical Engineering
Toyohashi University of Technology
Tempaku, Toyohashi 441-8580, Japan
E-mail: y-noda@tut.jp

***Abstract*—Logic circuit can be reduced, if any input is given as a constant. The derived circuit usually operates at a higher frequency, as the logic depth and layout area are reduced. Such technique is called as hardware specialization, since the circuit is specialized to specific input data. This study presents the quantitative evaluation results of hardware specialization for vibration control systems, using MATLAB/Simulink and Altera Cyclone III FPGA as evaluation platform. In case of the third order Butterworth low-pass filter, its logic scale was reduced to 40–56% of the original by fixing the gain parameters; the improvement of operational frequencies were 23–39%. In Hybrid Shaped Approach, the logic scale was reduced to 48% of the original by fixing the filter parameters, while the improvement of operational frequency was 43%. In Preshaping with a low-pass filter, the respective logic scales were reduced to 10.6%, 9.6%, and 1.2% of the original by fixing parameters, input data, and both. The respective improvements of operational frequencies were 53, 10, and 70%.**

***Keywords*—Digital Filter; Reconfigurable Hardware; Field Programmable Gate Arrays; Partial Evaluation**

I. Introduction

The demands for high-performance control systems are ever increasing. Although embedded processors have achieved remarkable improvement in performance, there are a number of applications whose requirements are difficult to be satisfied by a common embedded processor.

Hardware implementation of control logic has been studied as a prospective alternative to software-based systems. Hardwired logic generally outperforms the corresponding software by some orders of magnitude. Moreover, hard-wired logic results in less power consumption and smaller space factor in many cases.

An obvious drawback of hardwired control is the increase of design cost and implementation cost. However, most of these obstacles have been overcome by recent advances in computer and Field Programmable Gate Array (FPGA) technologies. FPGA is a reconfigurable LSI, which can be arbitrarily reprogrammed by downloading the configuration data. A recent FPGA device provides millions of logic gates at a reasonable cost, which is sufficient for most control applications. Logic design is now fully automated, and can be easily handled by a common personal computer. The burdens of hardware implementation are much alleviated by using FPGA devices.

It should be noted that each of hardware and software has its merits and demerits. Some portions of control are naturally suited for software implementation (e.g., user interface and network communication), and a practical system would be a hybrid of highly responsive hardware part and flexible software part. Although this study focuses on the implementation issues of hardware part, it never means that software part is negligible.

The purpose of this study is to evaluate the hardware specialization techniques for hardwired control logic. This study particularly examines digital filters and vibration control systems as simple examples of control applications. The rest of this paper is organized as follows. In Section II, the concept of hardware specialization is introduced with the related studies. Section III examines IIR digital filters that are frequently used in control applications. Then, two control applications are evaluated in Section IV. Section V concludes the paper.

II. Background and Related Studies

If the input of software is partially (or wholly) predetermined, the software can be optimized by using the predetermined values. For example, if a multiplicand is given as a constant, the corresponding multiplication can be replaced with the appropriate shift and add/sub operations to reduce execution time (strength reduction). If a branch condition is predetermined, the corresponding conditional branch instruction can be eliminated. If there is any code in the condition part that is never taken, such code can be also eliminated (dead code elimination). Such techniques are formally categorized as *Partial Evaluation* [1] [2], which transfers a generic program to the one that is specialized for a given set of input values.

Partial evaluation is also applicable to hardware design, which is referred to as *Hardware Specialization*. For example, if an input x of AND(x, y) is predetermined, the AND gate can be eliminated by replacing its output with 0 or y. The derived circuit is expected to be smaller in logic scale and higher in operational frequency.

Hardware specialization had not been fully exploited until FPGA technology became widespread. Since the specialized circuit has to be generated for each set of input values, it is

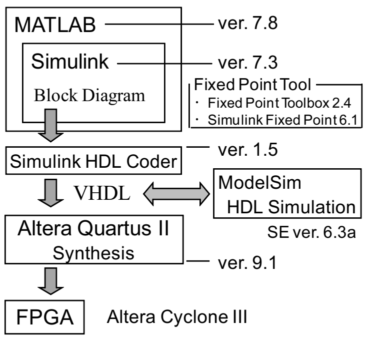

Fig. 1. Evaluation platform.

natural to use reconfigurable devices to reprogram the internal logic according to the input data.

To date, hardware specialization has been applied to some applications [3]: e.g., key-specific encryption circuit [4] [5]. Hardware specialization has long been acknowledged in signal processing applications, which frequently perform constant multiplications [6]. A well-known example is distributed arithmetic [7], which is a sophisticated implementation of the inner-product of constant coefficient vector and input vector.

This study focuses on quantitative evaluations of hardware specialization techniques for digital filters and control applications. Although the hardware specialization of digital filter is trivial in signal processing community, systematic and quantitative evaluation results have been scarcely reported on this topic. We barely found Chou et al. [8] reported that the logic scale of their IIR filter was reduced to 50% of the original by fixing the coefficients of the filter.

III. Evaluation of Digital Filters

A. Evaluation Platform

In the research and development of control systems, Simulink is widely used to design time-varying systems. Simulink is integrated with MATLAB, and provides an interactive graphical environment for model-based design, simulation, and implementation. Simulink HDL Coder is optional software to generate cycle-accurate HDL codes (Verilog HDL or VHDL) from Simulink models. The HDL codes generated by HDL coder are portable and synthesizable with most popular synthesis tools.

Figure 1 summarizes the evaluation process and the tools used in this study. First, the target model (continuous model) is constructed and verified with Simulink. The model is then discretized to construct the corresponding discrete model with adequate fixed-point data type. The sampling frequency is set to 512 Hz in all examples of this study. HDL coder converts this discrete model into VHDL codes, which are synthesized, placed, and routed with Altera Quartus II software. Target device is set to Altera Cyclone III FPGA EP3C120F780, while the optimization option is set to Balanced.

For verification purpose, some of the designs were actually downloaded into an Altera Cyclone II Starter Kit (EP3C25F324), which was driven by 25 MHz system clock. By checking the output signals written in on-board SRAM, our designs were proven to be fully functional.

TABLE I
Fixed-point data types for four forms.

	length	sign	integer	fraction
DF1	32	1	2	29
DF2	38	1	17	20
Cascade	32	1	11	20
Parallel	32	1	11	20

B. Design of Digital IIR Filters

In the following discussion, we examine a kind of low-pass filter, namely the third order Butterworth filter with cutoff frequency 3 Hz. Its transfer function is given by the following equation.

$$H(s) = \frac{6997.4}{1 + 37.6991s + 710.6115s^2 + 6997.4s^3} \qquad (1)$$

There are two categories of digital filter: infinite impulse response (IIR) and finite impulse response (FIR). Each of IIR and FIR filters has its own merits and demerits, and should be selected according to the requirements of each individual case. Though IIR filters are specifically examined in this study, our techniques are not specific to IIR filters and are equally applicable to FIR filters.

First, the filter given by Eq. (1) was designed and verified as an analog filter using Simulink. This analog model was then discretized via bilinear transformation with prewarping. Although there are many designs that realize a given transfer function, four typical forms were actually designed and evaluated. Figure 2 illustrates the block diagrams of four forms listed below.

- Direct Form I (**DF1**)
- Direct Form II (**DF2**)
- Biquad Cascade Form (**Cascade**)
- Biquad Parallel Form (**Parallel**)

We adopted fixed-point arithmetic, which is sufficient for most control applications and preferable for hardware implementation. The data types of four forms are summarized in Table I. We basically adopted 32-bit format for each form to equalize the evaluation condition. However, in DF2, we had to extend the width to 38 bit to avoid the loss of significant digits. Though it is possible to adopt different data types for each part of a digital filter, we applied the same data type to every part of a design in this work. The necessary width and the position of decimal point are also dependent on the specifications of input signal and filter characteristics; in this study, we selected a data type that is sufficient to handle a square wave (period 10 [s], amplitude 1, and 50% duty) for each filter design.

C. Hardware Specialization of IIR Digital Filters

For each of four forms shown in Fig. 2, the following two Simulink models are constructed and converted into VHDL codes, which are then synthesized and implemented

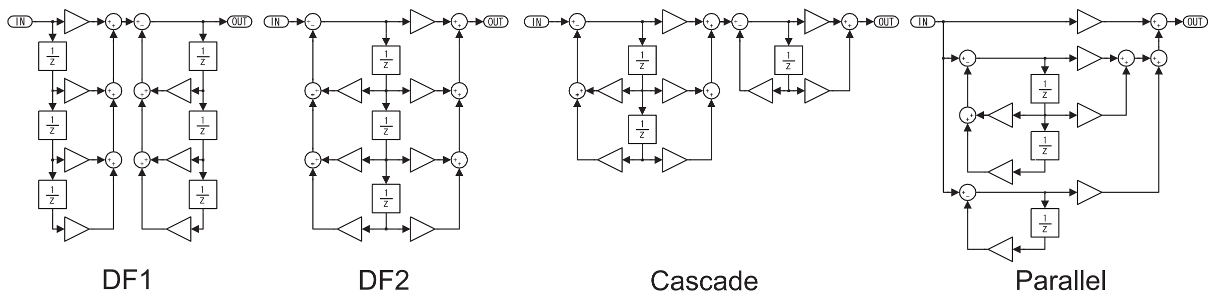

Fig. 2. Four forms of the third order Butterworth filter.

TABLE II
EVALUATION RESULTS OF FOUR FORMS OF THE IIR LOW-PASS FILTER.

	DF1		DF2		Cascade		Parallel	
	Specialized	Generic	Specialized	Generic	Specialized	Generic	Specialized	Generic
Total logic elements [LE]	686	1226	768	1939	585	1240	598	1109
Combinational Logics	558	1098	692	1863	521	1176	566	1077
Registers only	96	96	38	38	0	0	0	0
Combinational Logics with Register	32	32	38	38	64	64	32	32
Memory bits	0	0	0	0	0	0	0	0
Embedded Multiplier 9-bit elements	40	56	53	119	44	64	40	56
I/O Pins	68	292	80	346	68	324	68	292
Frequency [MHz] (85°C)	47.3	38.6	51.5	37.9	31.2	23.7	62.2	44.9
Circuit generation time [s]	107	133	132	162	114	170	94	115

for Cyclone III FPGA. The first one is **Generic**, which is the model whose filter parameters are given externally from input ports. The other is **Specialized**, which is the model whose filter parameters are given internally as constant values.

In Specialized design, filter coefficients are given as constants in the Simulink model, which is converted into the VHDL codes that include constant values. Hardware specialization is performed as a part of logic synthesis phase. It is worth considering designing dedicated software for hardware specialization of digital filters, though it is left for future studies.

Table II summarizes the evaluation results of four forms. Logic scale is expressed in logic elements (LEs) of Cyclone III FPGA. In each form, the logic scale of Specialized was reduced to approximately 50% of that of Generic. Although Generic DF2 has 38-bit data path and the largest in logic scale, 60% of its logic scale was reduced by hardware specialization. In any case, combinational logic is the dominant factor of logic scale, and the reduction of logic scale was mainly induced by reducing combinational logic gates. No reduction was observed in register elements. Operational frequencies were also improved by specialization in all forms; the improvement was between 23% (DF1) and 39% (Parallel).

The circuit generation time is another important factor in hardware specialization, since the specialized hardware has to be generated for each input data. Table II lists the circuit generation time of each design, where a common personal computer (Athlon 64 X2 4400+ with Windows XP SP3) was used. As readily seen, the circuit generation time is less than 3 minutes in these cases.

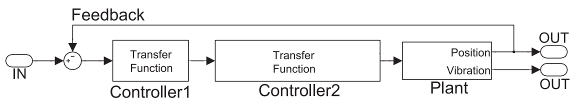

Fig. 3. Overview of Hybrid Shaped Approach.

IV. EVALUATION OF VIBRATION CONTROL SYSTEMS

A. Hybrid Shaped Approach (HSA)

Hybrid Shaped Approach (HSA) [9] [10] is a vibration control algorithm, which was originally designed for liquid container transfer. Figure 3 is a block diagram of a servo system that drives the target to follow the reference trajectory. This system contains two controllers and one target plant with a feedback path of target position. Controller 1 is a low-pass filter, whose transfer function is given by Eq. (2). Controller 2 is a notch filter (or band-stop filter) given by Eq. (3). The target Plant is a second order system given by Eq. (4), where the natural angular frequency $\omega_n = 1.937$ Hz and the damping factor $\zeta = 0.1$ are assumed.

$$H(s) = \frac{15.7}{1 + 0.005s} \tag{2}$$

$$H(s) = \frac{(2\pi \times 1.937)^2 + 2 \times 0.001 \times 2\pi \times 1.937s + s^2}{(2\pi \times 1.937)^2 + 2 \times 2\pi \times 1.937s + s^2} \tag{3}$$

$$G(s) = \frac{k\omega_n}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4}$$

Figure 4 illustrates the block diagram of HSA controller. FB_IN is the target position fed back from the target plant. Controller 1 and 2 are implemented as IIR filters of Cascade form. This model was then discretized for FPGA implementation with a 32-bit fixed-point format, which consists of sign

TABLE III
EVALUATION RESULTS OF HSA IMPLEMENTATIONS.

	Unfixed	Fixed Parameter	Fixed Input	All Fixed
Total LEs (utilization %)	1232 (1.0%)	587 (0.5%)	1277 (1.1%)	617 (0.5%)
Combinational Logics	1200	555	1212	552
Registers only	0	0	0	0
Combinational Logic with Register	32	32	65	65
Memory bits	0	0	0	0
Embedded Multiplier 9-bit elements	64 (11.1%)	36 (6.3%)	64 (11.1%)	36 (6.3%)
I/O Pins	356 (66.9%)	100 (18.8%)	324 (60.9%)	68 (12.8%)
Frequency [MHz] (0 – 85°C)	28.5 – 26.0	41.0 – 37.3	28.6 – 26.1	40.8 – 37.2
Circuit generation time [s]	139	99	131	91

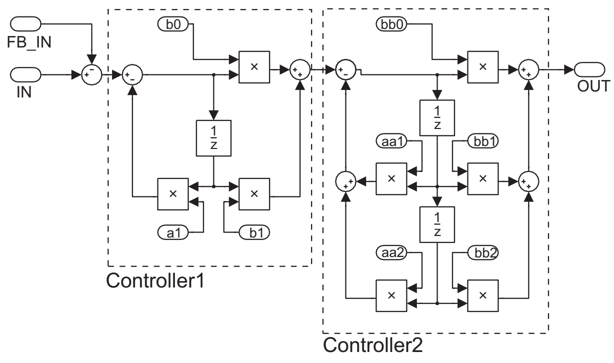

Fig. 4. Block diagram of the discretized HSA controller.

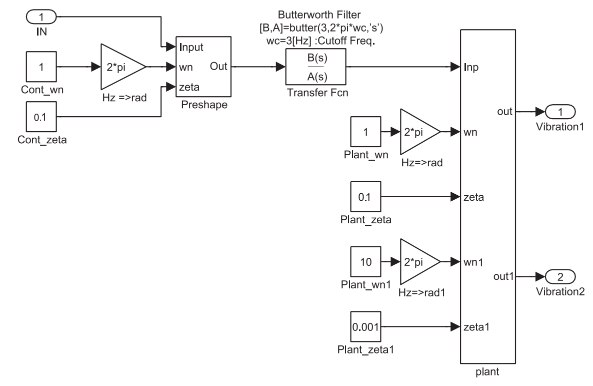

Fig. 5. Overview of Preshaping with a low-pass filter

(1 bit), integer part (15 bits), and fraction part (16 bits).

To examine the effect of hardware specialization, the following four designs were implemented and quantitatively evaluated.

- **Unfixed** is the model where filter parameters and input data are fed externally via input pins.
- **Fixed Parameter** is the model where all filter parameters are given as constants. Input data are fed externally.
- **Fixed Input** is the model where input data are generated internally. In this study, input data are assumed to be a square wave of period 10 [s], amplitude 1, and 50% duty. The pattern generator of this square wave is described in VHDL, and directory connected to the input of the controller. All filter parameters are fed externally via pins.
- **All Fixed** is the model, where all filter parameters are given as constants, while input data are given by the internal pattern generator.

To see the usages of these four designs, the vibration control of a transfer system is considered here as an example. The vibration of a transfer system depends on the natural frequency and the transfer path. Since the natural frequency is dependent on the target object, the filter parameters have to be adjusted to change the properties of the target object. Thus, the design Unfixed is used to transfer various objects via various paths. If the properties of target object are fixed, the design Fixed Parameter is enough to transfer the object via various paths. Similarly, Fixed Input is used to transfer various object via a fixed transfer path. All Fixed corresponds to the case where the properties of object and the transfer path are both fixed. Instead using All Fixed, it might be possible to output the precomputed values stored in a table; the problem of this approach is that the table sometimes becomes very large. Thus, All Fixed may be practically adopted to output very long sequence of data.

Table III summarizes the implementation results of four designs of HSA. The numbers in parenthesis are the resource utilization of an EP3C120F780 device. As readily seen, Fixed Parameter is 50% smaller and 40% faster than Unfixed. Meanwhile, Fixed Input is larger than Unfixed; no reduction by hardware specialization was observed, and the logic scale slightly increased by the input pattern generator (46 LE). All Fixed is similar to Fixed Parameter, while it is larger than Fixed Parameter by 30 LE.

B. *Preshaping with a Low-pass Filter*

Preshaping [11] is a feedforward control technique to reduce system vibration; it alters system inputs so that the system completes the requested move without residual vibration. For the second-order system of ω_n and ζ, it is possible to suppress the residual vibration by using the ΔT-delayed signal and the proper scaling factor K.

$$\Delta T = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \tag{5}$$

$$K = \exp\left(\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}\right) \tag{6}$$

Figure 5 illustrates the Simulink model of a simple vibration control system, whose target plant has two kinds of oscillations $(\omega_n, \zeta) = (1, 0.1)$ and $(10, 0.001)$. The controller consists of a Preshape block and a low-pass filter block. Preshape block

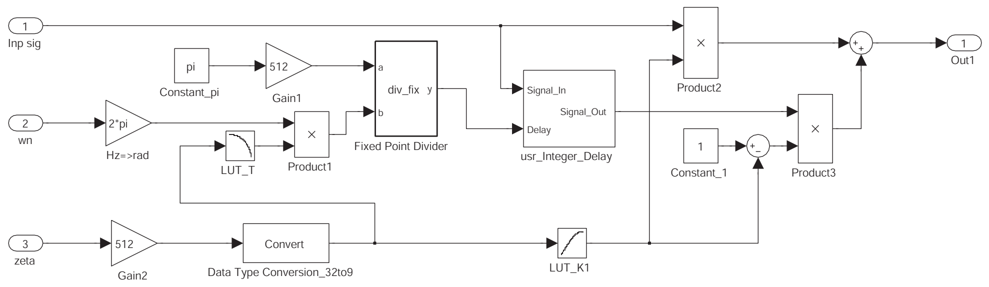

Fig. 6. Block diagram of a discretized Preshape block.

TABLE IV
EVALUATION RESULTS OF PRESHAPING IMPLEMENTATIONS.

	Unfixed	Fixed Parameter	Fixed Input	All Fixed
Total LEs (utilization %)	86271 (72%)	9114 (7.7%)	8305 (7.0%)	1071 (0.9%)
Combinational Logics	20703	858	6161	718
Registers only	36564	8192	1023	256
Combinational Logics with Register	29004	64	1121	97
Memory bits	0	0	0	0
Embedded Multiplier 9-bit elements	96 (17%)	60 (10%)	94 (16%)	58 (10%)
I/O Pins	388 (73%)	68 (13%)	356 (67%)	36 (7%)
Frequency [MHz] (0 – 85°C)	17.4 – 18.9	26.4 – 29.0	19.3 – 20.8	29.6 – 32.3
Circuit generation time [hour]	11.1	10.3	3.2	3.1

suppresses the vibration of low frequency by adjusting the input sequence for a given set of (ω_n, ζ). The low-pass filter block is the third order Butterworth filter shown in Section III, and restrains the vibration of high frequency.

This controller was then discretized with a 32-bit fixed-point data type which consists of sign (1 bit), integer part (11 bits), and fraction part (20 bits). The low-pass filter was implemented in Cascade form as shown in Fig. 2. The block diagram of Preshape block is shown in Fig. 6.

In implementing Preshape block, there are three major tasks: calculating K, calculating ΔT, and generating ΔT-delayed input signal. Since it is bulky to implement a square root function and an exponential function for ΔT and K with combinational logic, these functions were implemented as look-up tables.

First, Convert block converts the input ζ into 9-bit format, which is enough for this application. LUT_T is a look-up table of 512 entries, which outputs $\sqrt{1-\zeta^2}$ in 32-bit format. This value and ω_n are multiplied together to generate ΔT with a fixed-point divider (Eq. (5)). Since the HDL coder cannot handle the Divide block of Simulink directly, a simple MATLAB function was prepared to generate a "/" operator in HDL codes. An actual divider is generated by synthesis tools; LPM_DIVIDE was adopted in our case.

ΔT is then fed to the usr_Integer_Delay block to generate ΔT-delayed input signal. The usr_Integer_Delay block is described as an embedded MATLAB function, and is implemented as cascaded registers with an output multiplexer. The maximal delay time is set to 10 seconds; i.e., 5120 stages of registers are required. Consequently, this block consumes the majority of register resources of this controller.

Another look-up table (LUT_K1) outputs $1/(1+K)$, which is multiplied by the input signal; meanwhile, the ΔT-delayed input signal is multiplied by $K/(1+K) = 1 - 1/(1+K)$. The final output of Preshape block is the sum of these two products.

The following four designs were implemented and evaluated to examine the effects of hardware specialization. Filter parameters and Preshaping parameters are displayed in Fig 5. The input waveform was assumed to be a square wave of period 10 [s], amplitude 1, and 50% duty as in Section III.

- **Fixed Parameter** is the model whose Preshaping parameters and filter parameters are fixed as constants. The input waveform is given externally.
- **Fixed Input** is the model whose input waveform is generated internally. Preshaping parameters and filter parameters are given externally.
- **Unfixed** is the model whose Preshaping parameters, filter parameters, and input waveform are all given externally.
- **All Fixed** is the model whose Preshaping parameters and filter parameters are given as constants. The input waveform is generated internally.

Table IV lists the evaluation results of four designs. The target device and CAD options were set to the same ones as described in the previous section. The respective logic scales of Fixed Parameter and Fixed Inputs are 10.6% and 9.6% of Unfixed. Most of the reductions were derived from Preshape block, which is a dominant part of Unfixed. The operational frequencies were also improved.

It is interesting that the reduction of logic scale is much larger than in the cases of IIR filters. The reduction in Fixed Input is also remarkable, considering the increase in HSA. Even in Fixed Parameter, the reduction ratio of Preshaping is much larger than that in IIR filters. To investigate the reason of this phenomenon, we conducted another set of experiments.

Generally, logic synthesis software eliminates the logic

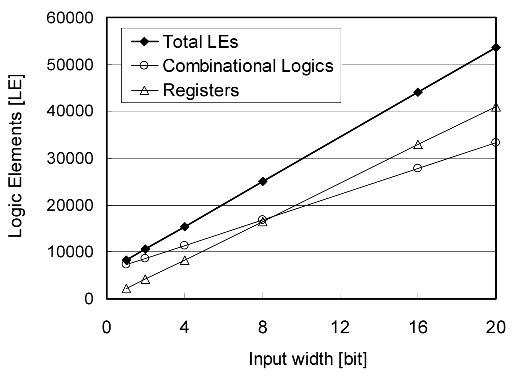

Fig. 7. The effect of input width on logic scale.

gates when there are any signals that have no effects on the output values. In the previous experiments, the input was a square wave which takes only two states. As a result, many bits of input data remain unchanged, which may lead to a drastic reduction in logic scale.

Hence we evaluated the logic scale of Fixed Input with a sawtooth wave generator, where a predefined number of bits change while the other bits are fixed to zero. Figure 7 summarizes the total number of LEs for various widths: 1, 2, 4, 8, 16, and 20 bit. (20 bit is the maximal width that can avoid the overflow in this computation.) The respective numbers of LEs for combinational logic gates and registers are also plotted in Fig. 7. The number of total LEs is not the sum of the corresponding numbers of combinatorial gates and registers. Each logic element (LE) contains a register element and a lookup table (for combinational logic), both of which can be used simultaneously.

As expected, the logic scale of Fixed Input was linearly dependent on the number of variable bits. Although the maximal reduction (90%) was derived with 1-bit sawtooth wave (i.e., a square wave), 38% reduction was still achieved with a 20-bit sawtooth wave, compared to Unfixed. Registers dominate the logic scale in a 20-bit case, while combinational logic gates dominate in a 1-bit case. It is natural that the number of registers is almost proportional to the number of variable bits, because the quantity of data to be held in usr_Integer_Delay is proportional to the number of variable bits in input data.

In summary, Fixed Input is effective to reduce register resources in this Preshaping hardware. Meanwhile, Fixed Parameter is generally effective to reduce combinational logic gates as shown in previous sections. All Fixed can thus achieve further reduction of logic scale by applying these two independent techniques simultaneously.

V. Conclusion

In this study, the effect of hardware specialization was quantitatively evaluated. In case of four forms of IIR digital low-pass filter, the average logic scale was reduced to 40%(DF2)–56%(DF1) of the original by fixing filter parameters. Similarly, the logic scales were reduced to 48% and 11% of the original by fixing filter parameters in HSA controller and Preshaping controller, respectively. In Preshaping, it was also possible to reduce the logic scale by fixing the input waveform; the logic scale was shown to be linearly dependent on the bit width that is necessary to represent the waveform of input data. The operational frequencies of specialized circuits were accordingly improved, as shown in Section IV-B.

By using hardware specialization, the implementation cost of control circuit can be significantly reduced; i.e., it is possible to implement more complex control system within the same cost of hardware. Hardware specialization is also suitable for the control applications that require very short response time, since it reduces the latency of the circuit. It is quite easy for users to apply hardware specialization techniques to their own designs. No special software is required, and users can work as usual with popular design tools: MATLAB, Simulink, HDL coder, and commercial FPGA design tools. Hardware specialization is particularly suited to FPGA technology, which has become a mainstream in embedded systems.

Currently, we are working on more complex and practical control systems that were difficult to be handled by software. There are still many items left for future studies.

Acknowledgment

This work was partially supported by JSPS KAKENHI (19500042).

References

[1] C. Consel and O. Danvy, "Tutorial notes on partial evaluation," in *Proc. 20th ACM Symp. on Principles of Programming Language*. ACM, 1993, pp. 493–501.

[2] N. D. Jones, "An introduction to partial evaluation," *ACM Computing Surveys*, vol. 28, no. 3, pp. 480–503, 1996.

[3] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, 2002, section 4.5 and 5.2.

[4] J. Leonard and W. H. Mangione-Smith, "A case study of partially evaluated hardware circuits: Key-specific DES," in *Proc. Field-Programmable Logic and Applications*, LNCS 1304. Springer, 1997, pp. 151–160.

[5] R. Atono et al., "Design and evaluation of data-dependent hardware for AES encryption algorithm," *IEICE Transactions on Information and Systems*, vol. E89-D, no. 7, pp. 2301–2305, 2006.

[6] R. Tessier and W. Burleson, "Reconfigurable computing for digital signal processing: A survey," *Journal of VLSI Signal Processing*, vol. 28, no. 1, pp. 7–27, 2001.

[7] S. A. White, "Applications of distributed arithmetic to digital signal processing: a tutorial review," *IEEE ASSP Magazine*, vol. 6, no. 3, pp. 4–19, July 1989.

[8] C.-J. Chou, S. Mohanakrishnan, and J. B. Evans, "FPGA implementation of digital filters," in *Proc. Fourth Intl. Conf. on Signal Processing Applications and Technology*, 1993, pp. 80–88.

[9] K. Yano, S. Higashikawa, and K. Terashima, "Liquid container transfer control on 3D transfer path by hybrid shaped approach," in *Proc. 2001 IEEE Intl. Conf. on Control Applications*, 2001, pp. 1168–1173.

[10] K. Yano, T. Toda, and K. Terashima, "Sloshing suppression control of automatic pouring robot by hybrid shape approach," in *Proc. 40th IEEE Conf. on Decision and Control 2001*, 2001, pp. 1328–1333, vol.2.

[11] N. C. Singer and W. P. Seering, "Preshaping command inputs to reduce system vibration," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 112, pp. 76–82, 1990.