Sensor Signal Processing Using High-Level Synthesis With a Layered Architecture

Hiroki Hihara, *Member, IEEE*, Akira Iwasaki, Masanori Hashimoto, *Senior Member, IEEE*, Hiroyuki Ochi, Yukio Mitsuyama, Hidetoshi Onodera, *Senior Member, IEEE*, Hiroyuki Kanbara, Kazutoshi Wakabayashi, *Member, IEEE*, Tadahiko Sugibayashi, Takashi Takenaka, Hiromitsu Hada, Munehiro Tada, *Senior Member, IEEE*, Makoto Miyamura, and Toshitsugu Sakamoto

***Abstract*—This letter describes a newly established design framework with the layered architecture of processing elements (PEs) exploiting high-level synthesis and its evaluation results. The design framework was developed for intelligent sensor nodes of Internet of Things applications that collaborate with cloud systems, in which small footprint and low power consumption were major concerns. The design framework consists of four layered structure of PE architecture with the extended database management function of a high-level synthesis tool. We investigated the dependencies of resource consumption on the granularity of coarse grained function definitions using the extended database management function of CyberWorkBench. The evaluation results showed that small footprint was achieved especially with dynamically reconfigurable technique.**

***Index Terms*—Behavioral synthesis, device computing, dynamically reconfigurable processor (DRP), field programmable gate array (FPGA), high-level synthesis, Internet of Things (IoT), micro controller.**

Manuscript received December 3, 2017; accepted January 11, 2018. Date of publication January 22, 2018; date of current version October 18, 2018. This manuscript was recommended for publication by P. S. Roop. *(Corresponding author: Hiroki Hihara.)*

H. Hihara is with the Research Center for Advanced Science and Technology, University of Tokyo, Tokyo 153-8904, Japan, and also with On-board Electronics Department, Space Engineering Division, NEC Space Technologies Ltd., Fuchu 183-8551, Japan (e-mail: hihara@sal.rcast.u-tokyo.ac.jp).

A. Iwasaki is with the Research Center for Advanced Science and Technology, University of Tokyo, Tokyo 153-8904, Japan (e-mail: aiwasaki@sal.rcast.u-tokyo.ac.jp).

M. Hashimoto is with Department of Information Systems Engineering, Graduate School of Information Science Technology, Osaka University, Suita 565-0871, Japan (e-mail: hasimoto@ist.osaka-u.ac.jp).

H. Ochi is with Department of Computer Science, College of Information Science and Engineering, Ritsumeikan University, Kusatsu 525-8577, Japan (e-mail: ochi@cs.ritsumei.ac.jp).

Y. Mitsuyama is with the School of Systems Engineering, Kochi University of Technology, Kami 782-8502, Japan (e-mail: mitsuyama.yukio@kochi-tech.ac.jp).

H. Onodera is with Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan (e-mail: onodera@i.kyoto-u.ac.jp).

H. Kanbara is with Research Division, Kyoto Advanced Laboratory, Kyoto 600-8813, Japan (e-mail: kanbara@astem.or.jp).

K. Wakabayashi and T. Takenaka are with Central Research Laboratories, NEC Corporation, Kawasaki 211-8666, Japan (e-mail: wakaba@bl.jp.nec.com; takenaka@aj.jp.nec.com).

T. Sugibayashi, H. Hada, M. Tada, M. Miyamura, and T. Sakamoto are with Central Research Laboratories, NEC Corporation, Tsukuba 305-8501, Japan (e-mail: t-sugibayashi@da.jp.nec.com; h-hada@bp.jp.nec.com; m-tada@bl.jp.nec.com; m-miyamura@aj.jp.nec.com; t-sakamoto@dp.jp.nec.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2018.2797064

I. Introduction

SENSOR nodes are important elements of Internet of Things (IoT) applications, whereas the transmission capacity of existing network channels is limited between sensors and cloud systems. In consequence, reducing data volume is necessary by accommodating device computing functions within sensor nodes to transmit sufficient data through existing networks. Small footprints and low power consumption are demanded for processing elements (PEs) within sensor nodes.

Field programmable gate arrays (FPGAs) are alternatives of microcontroller units (MCUs) to reduce the size of footprints, whereas the inefficiency of placement and routing (P&R) using configurable logic blocks and look-up tables (LUTs) with switches, together with associated configuration memories, is pointed out [1]. High-level synthesis is used for the design of FPGAs to enable global optimization and to compensate for the limitation of P&R.

We propose a new design framework with four-layer architecture to design embedded PEs in sensing devices of IoT applications using the database management function of a high-level synthesis tool. It exploits the repetitive high-level synthesis process. Macro blocks synthesized through high-level/behavioral synthesis are registered in a database before the system level synthesis, and the information in the database is used for the optimization of resource consumption through the system level synthesis. We investigated the dependencies of resource consumption on the granularity of coarse grained function definitions using the extended database management function of CyberWorkBench (CWB) [2]. The evaluation results show that small footprint was achieved especially with dynamically reconfigurable technique.

II. Related Work and Problem Formulation

Input sequences for embedded systems come from real world, and the input data for PEs are given serially. Backward data access is not necessary. Wire-rate processing using FPGAs is suitable for adding processing functions with minimum additional footprints.

Various optimization techniques of filtering functions for sensing images were summarized and proposed [3]–[5]. P&R as area-optimal mapping was also mentioned, whereas these improvements were based on the optimization of scheduling and did not exploit the functional representation of circuitry

definitions used in high-level synthesis. Design reuse of macro blocks by considering the functional representations of macro blocks used in high-level synthesis was proposed to optimize P&R process [6]. The reported method is applied to the layout process after logic synthesis. Code generation optimization using high-level synthesis was reported [7], whereas high-level synthesis is used for the optimization of streaming pipelines. If the functional representations of macro blocks are exploited in high-level/behavioral synthesis step, more chances of optimization can be found.

Dynamically reconfigurable processors (DRPs) using high-level synthesis were proposed [8]–[10] to improve the efficiency, whereas the inefficiency of fixed mesh pointed out in [1] still remains. Fixed bit widths of data paths, elementary blocks, and switch matrices aiming at mass production of the devices was one example of the inefficiency. Sensors used in IoT applications have various data interfaces, such as 8, 12, 14, or 16 bits. Predefined data path between arrays of arithmetic logic units (ALUs) prevents behavioral synthesis tools from the optimization of layout size and the reduction in power consumption. Therefore, optimized ALUs and flexible data paths are required to embed processors in sensor units.

We identified two issues those were missing in the conventional optimization methodologies. The first issue is the exploitation of the functional representation of macro blocks. The use of macro definitions in high-level synthesis using functional representations enables additional global optimization. The second issue is eliminating constraint found in conventional mesh type FPGAs and DRPs for the wire rate signal processings. Bit width of ALUs and data paths should be optimized for specific sensors while maintaining the merit of DRPs.

III. Architecture of Processing Elements

We established a new design framework to solve issues described in the preceding section by exploiting the binding process of high-level/behavioral synthesis.

A. Layered Architecture

The purpose of the conventional high-level synthesis tools is to generate finite state machines with data paths (FSMDs). We refer to FSMD as fine grained layer. We expanded the scope of high-level synthesis to coarse grained layer to exploit the functional representations of circuitries. ALUs in coarse grained layer are defined by the functional representations of high-level programming languages. We also added switches as bypass connection layer for the scope of high-level synthesis intentionally. Implicit as-built meshes of switches put constraints on high-level synthesis. We allowed some PEs to be just bypass switches to remove the meshes. Communications between the PEs are limited between adjacent PEs, whereas we found no limitations for sensor applications with the limited communication paths.

The four-layered structure is shown in Fig. 1. The layers consist of I/O circuitry, fine grained layer, coarse grained function definition layer, and bypass connection layer, where they are listed from the bottom layer to the top layer, respectively. The following explains these layers.

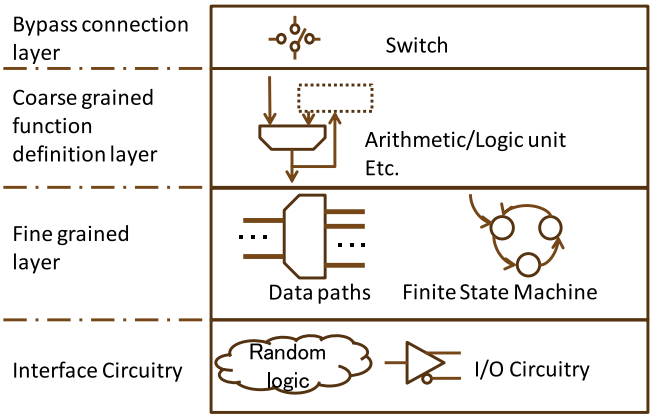

Fig. 1. Proposed four-layered structure.

I/O circuitry is implemented with random logic gates and mixed signal I/O circuitries. They are connected to the fine grained blocks in the above layer. The fine grained layer mainly consists of finite state machines and data paths. They are replaceable to follow the context described by high-level languages. The coarse grained function definition layer is located on the fine grained layer. Optimized ALUs for a certain application are defined in this layer. An operation primitive defined in the layer corresponds to an operation code (opcode) of a conventional MCU, whereas it does not have to be standardized for over many applications. As for the topmost bypass connection layer, simple bypass switch images can be specified over coarse grained blocks. The connections between inputs and outputs of PEs are configurable with the bypass switches.

B. Design Flow

The following six steps compose the design flow of the design framework to design a PE through layered scheme.

- Step 1: Describe a system in high-level language.
- Step 2: Define coarse grained operations. Typical dedicated functions for sensor signals are signal compensation, feature point identification for image recognition, and image compression in addition to basic arithmetic operations. The defined coarse grained operations are exploited in high-level synthesis and behavioral synthesis process.
- Step 3: Generate source codes written in hardware description language through behavioral synthesis. The hard macros defined and implemented in the step 2 are exploited for generating circuitries by the functions of CWB.
- Step 4: This step is identical to conventional logic synthesis.
- Step 5: This step is identical to conventional layout design.
- Step 6: The verification and validation step include back annotation based on the result of delay analysis.

We used a data base management function of CWB to exploit the definitions of operations in the coarse grained layer during the behavioral synthesis to treat a function as an operator. The operations can be implemented as hard macros by using custom ASIC design tools and/or LUTs of FPGAs. The

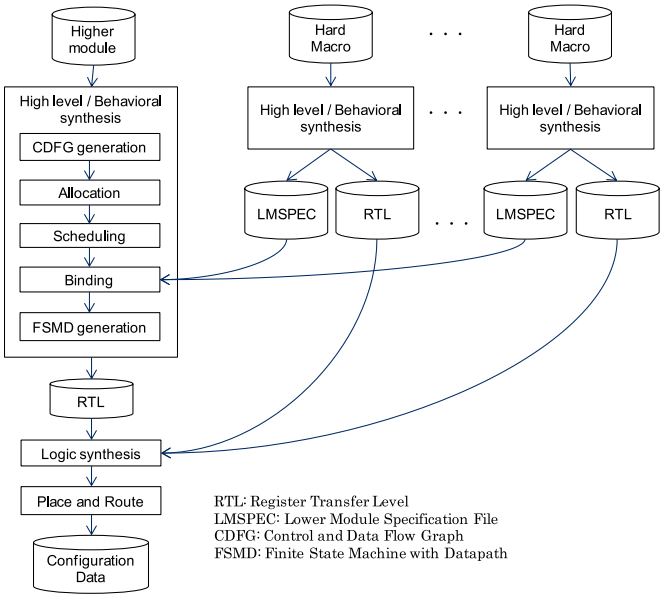

Fig. 2. Modified high-level synthesis process.

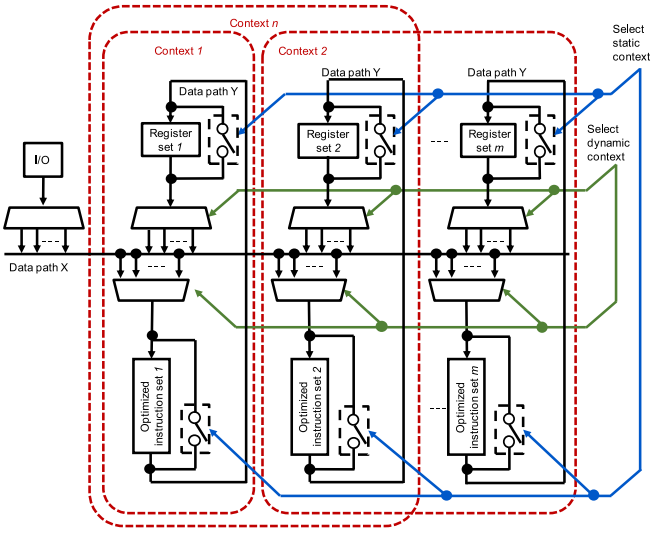

Fig. 3. Context implementation of a PE: On/Off state of switches are logical images.

operator definitions were exploited on the step 3 as macro blocks. They are registered in a database of CWB as shown in Fig. 2. Once macro blocks are registered in a database, CWB can use the macro blocks during the binding process of high-level synthesis automatically to reduce layout area size. The binding process of high-level synthesis is regarded as NP-hard and heuristic approach was often employed. The design flow enables the automation of binding process instead of the heuristic approach.

C. Typical Design of Processing Element

Two types of context were identified with reference to semantics or lambda calculus of functional representations to define ALUs in coarse grained function definition layer. One is a configurable static context often mentioned as stored information in a file, and the other is a dynamic context as stored information in heap registers as shown in Fig. 3. The static contexts of an application are expressed with finite state machines and data paths implemented by n sets of hardware circuitry of PEs. The dynamic contexts of an application are specified as m sets of registers and instructions. The instruction sets are optimized for an application, and each optimized instruction set can be shared among some dynamic contexts.

We implemented a specific C-language comment description /* Cyber func = operator */ for defining dynamic and static contexts, which are the sets of pairs of registers and optimized instruction sets. The pair of a register set and an instruction set can be shared among dynamic contexts using another specific C-language comment description as /* Cyber share_name = NAME */. NAME is an arbitrary designation. The binding process of high-level behavioral synthesis can be controlled by these descriptions to exploit the functional representations defined in a high-level programming language.

IV. Evaluations and Results

We evaluated the design framework using convolution operations, those are often used for sensor applications, with following three conditions. The matrix functions of the filters are similar and the difference is the parameters of 3×3 matrices. We could evaluate the layout area size reduction results by using an FPGA, Xilinx XC7A200T FPGA, although the design framework was established aiming at improving ASIC design at first.

1) Defining a convolution function as one operator.
2) Implementing ALUs with basic operations, i.e., plus, minus, multiply, divide, and comparison operations, using dynamically reconfiguration technique designated as flexible reliability reconfigurable array (FRRA) in [11].
3) Elaborating whole design with FPGA libraries without operator definitions and a function definition database.

We implemented a Laplacian filter with above three conditions in one case and the cascaded operations with a Gaussian filter and a Laplacian filter in another case. These filters are often used for image processing, and the cascaded operation is useful for robust edge detection. Operator definitions and resource sharing are specified for the condition 1) as shown in Fig. 4. Two filter functions were registered in the database by specifying CWB using the annotation "/* Cyber func = operator */," and they were used repeatedly as independent operators in the high-level and behavioral synthesis processes. The matrix functions were not registered independently in the database in the other conditions. The operator "+" can be shared between Laplacian_filter function and Gaussian_filter function by the annotation "/* Cyber sharename=ALU1 */." This annotation is a direction for CWB to assign the same operator for + in Laplacian_filter function and in Gaussian_filter function. If these functions are processed sequentially both + operation can be carried out by the same operator. On the other hand, if each + operator works in parallel and cannot be shared, a suffix is added to

```
/* Cyber func = operator */
unsigned short Laplacian_filter ( unsigned short pixel_buffer[PEL])
{
    short temp;
    pixel_buffer_s[9]; /* 3 x 3 convolution filter element */
   …
    temp = pixel_buffer_s[Num0] + /* Cyber share_name=ALU1 */
           pixle_buffer_s[Num1];
    …
}

/* Cyber func = operator */
unsigned short Gaussian_filter ( unsigned short pixel_buffer[PEL])
{
    short temp;
    pixel_buffer_s[9]; /* 3 x 3 convolution filter element */
   …
    temp = pixel_buffer_s[Num0] + /* Cyber share_name=ALU1 */
           pixle_buffer_s[Num1];
    …
}
```

Fig. 4. Operator definitions and operator sharing for condition 1).

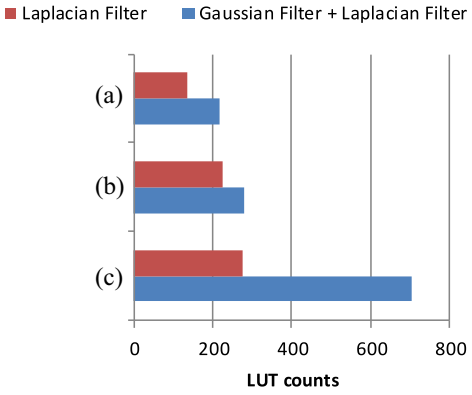

Fig. 5. Comparison of LUT utilizations according to operator definitions. (a) Using operator definition for a function. (b) Using operator definition with FRRA. (c) Using FPGA library without operator definitions.

the name ALU for independent identification and two operators are synthesized. This sharing status is decided by CWB automatically through the process of control and data flow analysis. Designers do not have to care about whether two functions are processed sequentially or in parallel, and they can concentrate on the analysis of the sharing possibility of operators using the annotation "/* Cyber func = …*/".

Derived LUT counts for each evaluation case are shown in Fig. 5. The LUT counts of basic operations were excluded for comparison. The larger the granularity of an operation was, the less counts of LUTs were consumed. Remarkable difference of LUT counts was shown for cascaded operations. The LUT counts of cascaded operations using FPGA library was more than double compared to single operation. The results show that exploiting granularity in behavioral synthesis was carried out by CWB automatically without specifying the reuse of macro blocks explicitly. The difference between one operation and cascaded operations using FRRAs with dynamically reconfiguration technique was 22% less than the difference using the operator definition on convolution operations.

V. Conclusion

We developed a novel design framework to reduce the footprints of programmable functions of sensing devices for IoT applications. The design framework consists of four layered structure of PE architecture and the extended database management function of a high-level synthesis tool to exploit functional representations in high-level programming languages.

We investigated the dependencies of layout size on the granularity of filtering function definitions using the extended database management function of CWB. We succeeded in automatic layout size reduction of an FPGA through the binding process of CWB by exploiting the functional representations in C language. It was realized by using annotations in program source code descriptions to show the information of resource sharing possibilities for high-level synthesis. This was achieved through step 2 and 3 of the design framework. We found that function definitions using dynamically reconfigurable processing technique is also effective to reduce layout size.

The layout size reduction is useful to embed a PE into a sensing device and to provide device computing capabilities with a sensing device. We also reported the power consumption reduction by adopting the design framework on a NanoBridge FPGA [12].

References

[1] R. Hartenstein, "The microprocessor is no longer general purpose: Why future reconfigurable platforms will win," in *Proc. 2nd Annu. IEEE Int. Conf. Innov. Syst. Silicon*, Austin, TX, USA, 1997, pp. 2–12.

[2] K. Wakabayashi, "CyberWorkBench: Integrated design environment based on C-based behavior synthesis and verification," in *Proc. IEEE VLSI-TSA Int. Symp.*, Hsinchu, Taiwan, 2005, pp. 173–176.

[3] R. Zhao, M. Tan, S. Dai, and Z. Zhang, "Area-efficient pipelining for FPGA-targeted high-level synthesis," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2015, pp. 1–6.

[4] F. Wang, G. T. Wang, R. H. Wang, and X. W. Huang, "FPGA implementation of Laplacian of Gaussian edge detection algorithm," *Adv. Mater. Res.*, vols. 282–283, pp. 157–160, Jul. 2011.

[5] L. Chen, "Fast generation of Gaussian and Laplacian image pyramids using an FPGA-based custom computing platform," M.S. thesis, Virginia Polytechnic Inst. State Univ., Blacksburg, VA, USA, 1994.

[6] M. Gort and J. Anderson, "Design re-use for compile time reduction in FPGA high-level synthesis flows," in *Proc. Int. Conf. Field Program. Technol. (FPT)*, Shanghai, China, 2014, pp. 4–11.

[7] M. Schmid, O. Reiche, C. Schmitt, F. Hannig, and J. Teich, "Code generation for high-level synthesis of multiresolution applications on FPGAs," in *Proc. FSP*, 2014, pp. 21–26.

[8] M. Meribout and M. Motomura, "A-combined approach to high-level synthesis for dynamically reconfigurable systems," *IEEE Trans. Comput.*, vol. 53, no. 12, pp. 1508–1522, Dec. 2004.

[9] T. Toi *et al.*, "High-level synthesis challenges and solutions for a dynamically reconfigurable processor," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, San Jose, CA, USA, 2006, pp. 702–708.

[10] Y. Hasegawa *et al.*, "Design methodology and trade-offs analysis for parameterized dynamically reconfigurable processor arrays," in *Proc. FPL*, Amsterdam, The Netherlands, 2007, pp. 796–799.

[11] D. Alnajjar *et al.*, "Reliability-configurable mixed-grained reconfigurable array supporting C-to-array mapping and its radiation testing," in *Proc. IEEE A-SSCC*, 2013, pp. 313–316.

[12] H. Hihara *et al.*, "Novel processor architecture for onboard infrared sensors," in *Proc. SPIE Infrared Remote Sens. Instrum. XXIV*, vol. 9973, San Diego, CA, USA, Sep. 2016, Art. no. 99730S.