Area/Delay Estimation for Digital Signal Processor Cores

Yuichiro Miyaoka[†] Yoshiharu Kataoka[†,*] Nozomu Togawa[††]
Masao Yanagisawa[†] Tatsuo Ohtsuki[†]

[†]Dept. of Electronics, Information and Communication Engineering, Waseda University
[††]Advanced Research Institute for Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku, Tokyo 169-8555, Japan
Tel: +81-3-5286-3396 Fax: +81-3-3203-9184
E-mail: miyaoka@ohtsuki.comm.waseda.ac.jp

Abstract— Hardware/software partitioning is one of the key processes in a hardware/software cosynthesis system for digital signal processor cores. In hardware/software partitioning, area and delay estimation of a processor core plays an important role since the hardware/software partitioning process must determine which part of a processor core should be realized by hardware units and which part should be realized by a sequence of instructions based on execution time of an input application program and area of a synthesized processor core. This paper proposes area and delay estimation equations for digital signal processor cores. For area estimation, we show that total area for a processor core can be derived from the sum of area for a processor kernel and area for additional hardware units. Area for a processor kernel can be mainly obtained by minimum area for a processor kernel and overheads for adding hardware units and registers. Area for a hardware unit can be mainly obtained by its type and operation bit width. For delay estimation, we show that critical path delay for a processor core can be derived from the delay of a hardware unit which is on the critical path in the processor core. Experimental results demonstrate that errors of area estimation are less than 2% and errors of delay estimation are less than 2ns when comparing estimated area and delay with logic-synthesized area and delay.

I. Introduction

A digital signal processor core is generally composed of a micro processor core and several hardware units for digital signal processing such as multiple memory buses, addressing units, and hardware loop units [7], [8]. However, if a particular application program runs on a general digital signal processor, some hardware units can be often used and other hardware units can never be used. We consider that an appropriate configuration for digital signal processor cores is required depending on the requirements for a given application program as well as hardware cost for a processor core. Hardware/software codesign can be one of the powerful design methodologies in order to obtain an appropriate configuration for processor cores. Several hardware/software codesign systems for processor core design have been reported such as in [1]–[3], [6].

We have been developing a hardware/software cosynthesis system for digital signal processor cores [9], [10]. Given an application program written in C and a set of application data, the system synthesizes a hardware description of a processor core. It also generates an object code and a software environment (a compiler and a simulator) for the processor core. In the system, one of the most important processes is hardware/software partitioning. In hardware/software partitioning, the system determines which part of a processor core should be realized by hardware units and which part should be realized by a sequence of instructions based on execution time of an input application program and area of a synthesized processor core. Execution time of an application program can be derived from critical path delay of a processor core and the number of clock cycles to run an application program. Thus we require area and delay estimation of a processor core in hardware/software partitioning. Area estimation has been discussed in [4], [6]. They first logic-synthesize each of hardware units and then they obtain estimated area by adding area for hardware units used in a processor core. However, this approach does not consider controlling area for hardware units in a processor core. We consider that we must take into account controlling area for hardware units in our system since it has many types of hardware units. Delay estimation has not been discussed so far and then we must establish a delay estimation technique for our system.

Based on the above discussion, we propose in this paper area and delay estimation equations for digital signal processor cores, which are incorporated into our hardware/software cosynthesis system. For area estimation, we show that total area for a processor core can be derived from the sum of area for a processor kernel and area for additional hardware units. Area for a processor kernel can be mainly obtained by minimum area for a processor kernel and overheads for adding hardware units and registers. Area for a hardware unit can be mainly obtained by its type and operation bit width. For delay estimation, we show that critical path delay for a processor core can be derived from the delay of a hardware unit which is on the critical path in the processor core.

This paper is organized as follows: Section II defines our processor model and its hardware parameters; Sections III and IV proposes area and delay estimation equations for digital signal processor core; Section V shows experimental results and evaluates the proposed area and delay estimation equations; Section VI gives concluding remarks;

II. Processor Model and Area/Delay Estimation Approach

In this section, we first define an architecture model of our processor core with its hardware parameters. The processor architecture model in this section is used in our hard-

[*] Presently, with Matsushita Communication Industrial Co., Ltd.

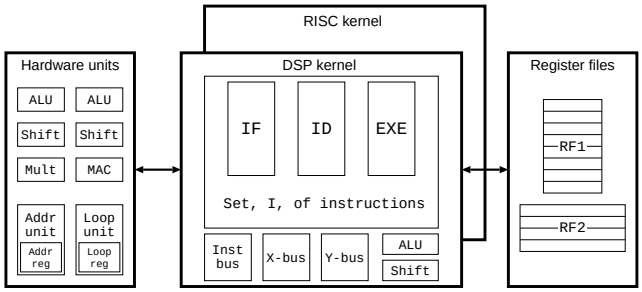

Fig. 1. Processor core configuration. Our processor core is composed of one of the two processor kernels, register files, and hardware units.

ware/software cosynthesis system [9], [10]. Then we discuss area and delay estimation approach.

Fig. 1 shows an architecture model of our processor core. The architecture model in Fig. 1 is based on the digital signal processor in [7], [8]. A *processor core* is composed of one of the two *processor kernels*, one or two *register files*, and several *hardware units*. In the following, processor kernels, register files, and hardware units are defined.

A. Processor Kernel

We have two types of processor kernels; (i) a RISC-type kernel and (ii) a DSP-type kernel. One of them is selected depending on a given application program by our hardware/software cosynthesis system. A RISC-type kernel has the five pipeline stages (IF, ID, EXE, MEM, and WB) as in the micro processor of [5]. A DSP-type kernel has the three pipeline stages (IF, ID, and EXE) as in the DSP processor of [7], [8]. The number of pipeline stages and processes in each pipeline stage are fixed and cannot be changed. A processor core will be a general-purpose RISC core if a RISC-type kernel is selected. It will be a DSP core if a DSP-type kernel is selected.

Each processor kernel has a Harvard architecture and consists of (c-i) a bus for an instruction memory, (c-ii) a bus for an X data memory (X-bus), and (c-iii) an ALU (Arithmetic Logic Unit) and a barrel shifter. In addition to these (c-i)–(c-iii), (c-iv) a bus for an Y data memory (Y-bus) is optionally added to a kernel. Data bus width of the instruction memory, the X data memory, and the Y data memory can be changed but their address bus width is fixed to 16 bits. The data bus width of the X data memory must be the same as the bit width of the Y data memory. Data bus width of the instruction memory can be determined based on a set of instructions included in a processor kernel.

Instructions in our processor kernel are grouped into *basic instructions* such as ADD and MUL and *parallel instructions* such as (ADD || ADD) and (ADD || MUL). The basic instructions correspond to the functions of our processor kernels and hardware units. A parallel instruction executes more than one basic instructions.[1]

[1] All the combination of basic instructions cannot be a parallel instruction but the hardware/software cosynthesis system determines which combination of basic instructions should be a parallel instruction.

The hardware parameters for a processor kernel are a kernel type t_{knl} (t_{knl} will be RISC or DSP), basic bit width b_{knl} for a kernel, basic bit width $b_{knl,fu}$ for the ALU and shifter, the number n_{bank} of data memory banks ($n_{bank} = 1$ or $n_{bank} = 2$), the number n_p of instructions executed concurrently, data bus width b_{data} and b_{inst} of the data memory and the instruction memory, and a set I of instructions in a processor kernel. t_{knl} determines a processor kernel type. b_{knl} determines bit width of pipeline registers. If $n_{bank} = 1$, the X data memory is used only. If $n_{bank} = 2$, both the X data memory and the Y data memory are used. n_p determines the maximum number of instructions executed concurrently. n_p is called a *parallel factor*.

B. Register Files

We have two types of register files; (i) a register file RF_1 and (ii) a register file RF_2. The register file RF_1 in Fig. 1 is used for all the instructions including arithmetic operations, logical operations, and address operations. The register file RF_2 is optionally added depending on a given application program by our hardware/software cosynthesis system and its bit width is longer than that of RF_1. RF_2 is used to store intermediate results for multiplication.

The hardware parameters for register files are the number n_{r1} and bit width b_{r1} of registers in the register file RF_1 and the number n_{r2} and bit width b_{r2} of registers in the register file RF_2. If $n_{r2} = 0$, the register file RF_2 is not added to a processor kernel and then the processor kernel has a single register file RF_1.

C. Hardware Units

Our processor core can have hardware units of (1) functional units (shifters, ALUs, multipliers, and MAC (multiply and add) units), (2) addressing units, and (3) hardware loop units.[2] All these hardware units (1)–(3) can be added to the DSP kernel. Only the hardware unit (1), i.e., functional units can be added to the RISC kernel.

The hardware parameter for hardware units is a set HU of hardware units which are added to a processor kernel. If $hu \in HU$ is a functional unit, it has parameters of basic bit width b_{fu} for operations. For example, a 16-bit ALU has basic bit width of 16 bits ($b_{fu} = 16$ for the 16-bit ALU). If $hu \in HU$ is an addressing unit, it has parameters of an addressing unit type t_{addr} ($t_{addr} = 2$ or $t_{addr} = 3$),[3] the number n_{dp} of address registers, and the number n_{dn} of index registers. If $hu \in HU$ is a hardware loop unit, it has a parameter of the number n_{loop} of loop registers.

[2] See [9], [10] for detailed operations of each hardware unit. Note that our processor kernel always includes an ALU and a shifter. HU in this subsection refers to a set of hardware units other than the ALU and shifter in a kernel.

[3] If $t_{addr} = 2$, an addressing unit has post increment/decrement operation. If $t_{addr} = 3$, an addressing unit has index add operation as well as post increment/decrement operation. Note that an address register and an index register has bit width of 16 bits since address bus width is fixed to 16 bits.

D. Area and Delay Estimation Approach

Given an application program and a set of application data, our hardware/software cosynthesis system for digital signal processor cores [9], [10] tries various sets of the hardware parameters described in Section II and determines each of the hardware parameters for a processor core which optimizes processor core area as well as execution time of the application program. In this optimization, the system requires area and delay estimation for a given set of hardware parameters. Thus we establish area and delay estimation equations of a processor core for a set of hardware parameters. Here we estimate logic-synthesized area and delay without really logic-synthesizing a processor core.

In order to establish area and delay estimation equations of a processor core, we first generate a variety of processor cores using our hardware/software cosynthesis system by varying a set of the hardware parameters. Processor cores are written in VHDL. Then we logic-synthesize these processor cores. We use Synopsys Design Compiler[4] as a logic synthesizer with the VDEC cell libraries (CMOS and 0.35μm technology).[5] We finally analyze the logic-synthesized area and delay of processor cores and establish their estimation equations based on their hardware parameters.

Based on this approach, we propose area and delay estimation equations of a processor core in the subsequent sections.

III. Processor Core Area Estimation

Our processor architecture model indicates that area for a processor core can be computed by adding up area for a processor kernel, area for register files, and area for hardware units. Thus Sections A–C discuss area for a processor kernel, area for register files, and area for hardware units, respectively. Then Section D summarizes area estimation equations.

A. Area Estimation for a Processor Kernel

A processor kernel is determined by a kernel type t_{knl}, basic bit width b_{knl} for a kernel, basic bit width $b_{knl,fu}$ for the ALU and shifter, the number n_{bank} of data memories, a parallel factor n_p, data bus width b_{inst} and b_{data} of the instruction memory and the data memory, and a set I of instructions. A kernel area is dependent on all of them. It is also dependent on register files and a set HU of hardware units. For simplicity, we assume that $b_{knl} = 32$, $b_{knl,fu} = 16$, and $b_{data} = 16$. We also assume that bit width b_{r1} and b_{r2} of register files RF_1 and RF_2 are 16 bits and 32 bits, respectively. These assumptions can be applied to typical digital signal applications for our hardware/software cosynthesis system.

Thus for given t_{knl} and n_p, we first analyze the relation between kernel area and the rest of hardware parameters including register file and hardware unit configuration. Then we establish area estimation equations for a processor kernel.

[4] We set the parameter map_effort of Design Compiler to be *mid*.

[5] The libraries in this study have been developed in the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration by Hitachi Ltd. and Dai Nippon Printing Corporation.

TABLE I
Instruction bit width and IF stage area.

Instruction bit width (b_{inst})	IF stage area (μm^2)
16	12543
24	15703
32	19440
48	24920
64	32750

A.1. Relation between Area for Processor Kernel and Hardware Parameters

Kernel area and instructions: An instruction is first fetched at the IF stage. Since bit width of each instruction is the same as b_{inst}, we consider that IF stage area is increased as b_{inst} is increased. Table I shows the relation between b_{inst} and logic-synthesized IF stage area. We observe that IF stage area is linearly increased as b_{inst} is increased.

A fetched instruction is decoded at the ID stage. In the ID stage, the decoder area is increased if the number of instructions, $|I|$, is increased. From the similar experiments to IF stage area, we observed that ID stage area is linearly increased as $|I|$ is increased.

Since the DSP-type kernel and RISC-type kernel have the same hardware for the IF stage and ID stage, the above observations are independent of a kernel type. They are also independent of a parallel factor n_p. Thus we have the following observation:

Observation 1 *Area for the IF stage in a processor kernel is linearly increased as b_{inst} is increased. Area for the ID stage in a processor kernel is linearly increased as $|I|$ is increased.*

Kernel area and data memory banks: If the Y-bus is added to a DSP-type kernel, the hardware for the ID stage and EXE stage is changed. If it is added to a RISC-type kernel, the hardware for the ID stage and MEM stage is changed. Kernel area is increased by the hardware for the Y-bus. From the experiments, we observed that the amount of the increased kernel area is independent of n_p. The hardware for the Y-bus is independent of other parameterized hardwares. Thus we have the following observation:

Observation 2 *Kernel area is increased if Y-bus is added depending on a kernel type.*

Kernel area and register files: Let b_{opr} be bit width of an operand field of an instruction.[6] b_{opr} must be greater or equal to $\lg(n_{r1} + n_{r2})$, where n_{r1} and n_{r2} are the numbers of registers in the register files RF_1 and RF_2, respectively. If the number of registers in each register file is increased, b_{opr} must be increased and then it must cause area increase in a processor kernel.

Let us assume a processor kernel where $t_{knl} = \text{DSP}$, $n_p = 1$, $n_{bank} = 1$, $b_{opr} = 2$. In this case, if b_{opr} is increased, we have the amount of increased kernel area compared with the assumed processor core shown in Table II. The amount of increased kernel area is linearly increased as b_{opr} is increased. From further experiments, we observed that it is independent

[6] In our processor core, a basic instruction has $n_p \times 3$ operand fields.

TABLE II
BIT WIDTH OF AN OPERAND FIELD AND THE AMOUNT OF INCREASED KERNEL AREA.

Bit width of an operand field (b_{opr})	Amount of increased kernel area (μm^2)
3	2313
4	5741
5	8317

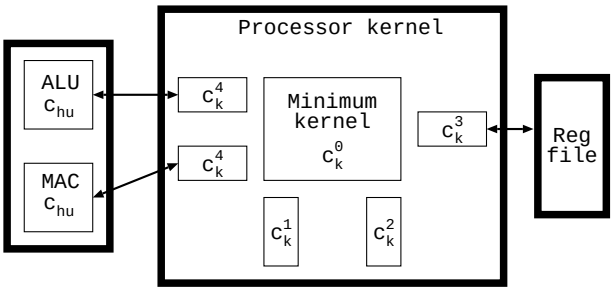

Fig. 2. Processor core estimation using a minimum processor kernel.

of hardware unit configuration but dependent on t_{knl} and n_p. Thus we have the following observation:

Observation 3 *The amount of increased kernel area is linearly increased as b_{opr} is increased. It is independent of hardware units but dependent on t_{knl} and n_p.*

Kernel area and hardware units: If a hardware unit *hu* is added to a processor kernel, the hardware for the EXE stage is changed and a controller for *hu* is added to the EXE stage. Since the hardware for the controller is independent of other hardwares, we can estimate an amount of increased kernel area for each hardware unit controller. It is independent of t_{knl} and n_p. Thus we have the following observation:

Observation 4 *The amount of increased kernel area for a hardware unit hu is dependent on hu and independent of other hardware parameters.*

A.2. Area Estimation Equations for a Processor Kernel

First we define a minimum processor kernel *MP* (see Fig. 2). For *MP*, $b_{knl} = 32$ for the kernel basic bit width and $b_{knl,fu} = 16$ for the ALU and shifter in a kernel. t_{knl} can be DSP or RISC and n_p can be 1, 2, or 4. b_{opr} and b_{inst} are defined as 2 and 0, respectively. We assume that *MP* does not include any instructions. By logic-synthesizing a minimum processor kernel for various t_{knl} and n_p, we have area $c_k^0(t_{knl}, n_p)$ for minimum processor kernel as follows:

$$\begin{aligned} c_k^0(RISC,1) &= 584033 \\ c_k^0(RISC,2) &= 698268 \\ c_k^0(RISC,4) &= 1116272 \\ c_k^0(DSP,1) &= 488137 \\ c_k^0(DSP,2) &= 565726 \\ c_k^0(DSP,4) &= 770367 \end{aligned} \quad (1)$$

Based on Observation 1, we have the amount of increased kernel area, $c_k^1(b_{inst}, I)$, for a set of instructions as follows:

$$c_k^1(b_{inst}, I) = [444 \times b_{inst} + 5439] + 361 \times |I| \quad (2)$$

The first term shows the IF stage area and the second term shows area increase for the number of instructions.

TABLE III
THE AMOUNT OF INCREASED AREA, $c_k^4(hu)$, FOR A HARDWARE UNIT $hu \in HU$.

Hardware unit		$c_k^4(hu)$
Functional unit	ALU (16 bits)	12736
	SHIFT (16 bits)	13418
	ADD (16 bits)	15689
	ALU (32 bits)	12736
	SHIFT (32 bits)	13418
	ADD (32 bits)	15689
	MULT(16 bits)	13345
	MAC (16 and 32 bits)	$48685 + 3639 \times n_{r1} + 5554 \times n_{r2}$
Addressing unit	$t_{addr} = 2$	20267
	$t_{addr} = 3$	19412
Hardware loop unit		10300

Based on Observation 2, we have the amount of increased kernel area, $c_k^2(t_{knl}, n_{bank})$, for data memory banks as follows:

$$\begin{aligned} c_k^2(RISC, n_{bank}) &= 13220 \times (n_{bank} - 1) \\ c_k^2(DSP, n_{bank}) &= 12591 \times (n_{bank} - 1) \end{aligned} \quad (3)$$

Based on Observation 3, we have the amount of increased kernel area, $c_k^3(t_{knl}, n_p, b_{opr})$, for register files as follows:

$$\begin{aligned} c_k^3(RISC,1,b_{opr}) &= 3231 \times (b_{opr} - 2) \\ c_k^3(RISC,2,b_{opr}) &= 6963 \times (b_{opr} - 2) \\ c_k^3(RISC,4,b_{opr}) &= 13088 \times (b_{opr} - 2) \\ c_k^3(DSP,1,b_{opr}) &= 2952 \times (b_{opr} - 2) \\ c_k^3(DSP,2,b_{opr}) &= 6123 \times (b_{opr} - 2) \\ c_k^3(DSP,4,b_{opr}) &= 12100 \times (b_{opr} - 2) \end{aligned} \quad (4)$$

Based on Observation 4, we have the amount of increased area, $c_k^4(hu)$, for a hardware unit $hu \in HU$ as shown in Table III.

Finally processor kernel area c_k is estimated as follows:

$$\begin{aligned} c_k &= c_k^0(t_{knl}, n_p) + c_k^1(b_{inst}, I) \\ &\quad + c_k^2(t_{knl}, n_{bank}) + c_k^3(t_{knl}, n_p, b_{opr}) \\ &\quad + \sum_{hu \in HU} c_k^4(hu) \end{aligned} \quad (5)$$

B. Area Estimation for Register Files

Area for a register file is increased if the number of registers in the register file is increased. Area for a register file is also increased if n_p is increased. This is because the register file requires extra input/output ports to write/read registers. Based on this discussion, we logic-synthesized various register files. Then given bit width b_r and the number n_r of registers in a register file, we obtained the following equations $c_r(b_r, n_r, n_p)$ for area estimation for the register file:

$$\begin{aligned} c_r(16,n_r,1) &= 12269 \times n_r + 28992 \\ c_r(16,n_r,2) &= 21433 \times n_r + 29037 \\ c_r(16,n_r,4) &= 36472 \times n_r + 153446 \\ c_r(32,n_r,1) &= 26405 \times n_r - 8184 \\ c_r(32,n_r,2) &= 40422 \times n_r - 4086 \\ c_r(32,n_r,4) &= 67184 \times n_r + 5521 \end{aligned} \quad (6)$$

Since our processor core has two register files RF_1 and RF_2, area c_{rf} for these register files is estimated as:

$$c_{rf} = c_r(16, n_{r1}, n_p) + c_r(32, n_{r2}, n_p) \quad (7)$$

TABLE IV
HARDWARE UNIT LIBRARY.

Hardware unit		Area [μm^2]	Delay [ns]
Functional unit	ALU (16 bits)	78915	2.82
	SHIFT(16 bits)	38859	1.02
	ADD (16 bits)	25259	1.44
	ALU (32 bits)	151194	3.25
	SHIFT (32 bits)	96838	1.35
	ADD (32 bits)	41963	2.49
	MULT(16 bits)	356948	5.71
	MAC (16 and 32 bits)	463716	7.28
Addressing unit	$t_{addr} = 2$ (no registers)	20154	—
	$t_{addr} = 3$ (no registers)	30535	—
Hardware loop unit		156344	—

C. Area Estimation for Hardware Units

A hardware unit library is a set of hardware units, each of which is logic-synthesized and optimized by Synopsys Design Complier. Table IV shows our hardware unit library. Then hardware unit area $c_{hu}(hu)$ for each hardware unit $hu \in HU$ is estimated as following.

For area estimation of a functional unit, we directly use area in Table IV.

An addressing unit has the parameters of type t_{addr}, the number n_{dp} of address registers, and the number n_{dn} of index registers. If $n_{dp} = n_{dn} = 0$, Table IV gives an addressing unit area. Area $c_{r,addr}$ for address registers and area $c_{r,idx}$ for index registers can be estimated as:

$$\begin{aligned} c_{r,addr}(n_{dp}) &= c_{r,idx}(n_{dn}) \\ &= 11585 \times n_{dp(dn)} + 4006 \end{aligned} \quad (8)$$

Addressing unit area is estimated by adding up area for an addressing unit in Table IV and area for address registers and index registers.

A hardware loop unit has the parameter of the number n_{loop} of loop registers. If $n_{loop} = 0$, Table IV gives a hardware loop unit area. Area $c_{r,loop}$ for loop registers can be estimated as:

$$c_{r,loop}(n_{loop}) = 32004 \times n_{loop} + 12801 \quad (9)$$

Hardware loop unit area is estimated by adding up area for a hardware loop unit in Table IV and area for loop registers.

D. Area Estimation Equation for a Processor Core

Based on the above discussions, processor core area c [μm^2] is estimated as:

$$c = c_k + c_{rf} + \sum_{hu \in HU} c_{hu}(hu) \quad (10)$$

IV. Processor Core Delay Estimation

If a processor core has a RISC-type kernel, critical path delay d can be estimated by

$$d = \max\{d_{IF}, d_{ID}, d_{EXE}, d_{MEM}, d_{WB}\}, \quad (11)$$

where d_{IF}, d_{ID}, d_{EXE}, d_{MEM}, and d_{WB} are delays for the IF stage, ID stage, EXE stage, MEM stage, and WB stage, respectively. Similarly, if a processor core has a DSP-type kernel, a critical path delay d can be estimated by

$$d = \max\{d_{IF}, d_{ID}, d_{EXE}\}. \quad (12)$$

Since the EXE stage has almost all the functional units, d_{EXE} usually gives the maximum delay for both a RISC-type kernel and a DSP-type kernel, i.e., d will be equal to d_{EXE}. In this section we focus on EXE stage delay.

TABLE V
DELAY FOR THE REGISTER FILE RF_1.

# of registers (n_{r1})	Delay [ns]		
	$n_p = 1$	$n_p = 2$	$n_p = 4$
4	0.97	1.24	2.63
8	1.20	1.25	2.50
16	1.31	1.54	2.80
32	1.75	1.94	2.93

For a DSP-type kernel, EXE stage delay d_{EXE} is composed of functional unit delay d_{op}, controller delay d_{ctl} for functional units, and delay d_{wb} for writing back registers. d_{EXE} can be written as $d_{EXE} = d_{op} + d_{ctl} + d_{wb}$. For a RISC-type kernel, EXE stage delay d_{EXE} is written as $d_{EXE} = d_{op} + d_{ctl}$ since it has the WB stage which is independent of the EXE stage for writing back registers.

Functional unit delay: Functional unit delay d_{op} can be estimated based on Table IV. The delay column in Table IV gives delay $d(fu)$ for each functional unit fu. Then d_{op} is estimated by the maximum delay for all the functional units in a processor core.

Controller delay: From the experiments, controller delay d_{ctl} for a RISC-type kernel ranges from 1.70ns to 2.90ns. Controller delay d_{ctl} for a DSP-type kernel ranges from 1.50ns to 2.90ns. They are independent of other hardware parameters. Thus we estimate d_{ctl} as 2.90.

Register write delay: In typical digital signal processing applications, we can assume that $n_{r1} > n_{r2}$, i.e., the register file RF_1 has more registers than the register file RF_2. Then we consider writing delay for the register file RF_1 as the register writing delay.

Table V shows the writing delay for RF_1 for each parallel factor n_p. For each n_p, delay for RF_1 is increased as the number of registers is increased. Assume that our processor core has at most 32 registers in RF_1. Then we estimate d_{wb} as 1.75 for $n_p = 1$, 1.94 for $n_p = 2$, and 2.93 for $n_p = 4$.

Based on the above discussions, we estimate processor core delay $d(t_{knl}, n_p)$ [ns] as:

$$\begin{aligned} d(RISC, n_p) &= d_{op} + 2.90 \\ d(DSP, 1) &= d_{op} + 2.90 + 1.75 \\ d(DSP, 2) &= d_{op} + 2.90 + 1.94 \\ d(DSP, 4) &= d_{op} + 2.90 + 2.93 \end{aligned} \quad (13)$$

where d_{op} is functional unit delay.

V. Experimental Results

In order to verify the established area and delay estimation equations for a processor core, we generated several processor cores using our hardware/software cosynthesis system and logic-synthesized them using Synopsys Design Compiler. Table VI shows sets of hardware parameters for generated processor cores.

Tables VII and VIII shows the comparison results. These tables indicate that errors of area estimation are less than 2%

TABLE VI
HARDWARE PARAMETERS FOR EXPERIMENTED PROCESSOR CORES.

Processor id	1	2	3	4	5	6	7
Parallel factor n_p	1	1	1	1	1	2	2
Kernel type t_{knl}	RISC	RISC	DSP	DSP	DSP	DSP	DSP
Bit width of $RF1$ b_{r1}	16	16	16	16	16	16	16
Bit width of $RF2$ b_{r2}	32	32	32	32	32	32	32
Bit width of instructions b_{inst}	32	32	32	32	32	32	32
# of registers in $RF1$ n_{b1}	8	8	3	6	5	14	8
# of registers in $RF2$ n_{b2}	0	2	1	2	1	2	4
# of instructions $\|I\|$	34	35	21	21	23	23	32
# of data memories n_{bank}	1	1	1	1	1	2	2
# of ALU (16 bits)	1	1	1	1	1	1	2
# of SHIFT (16 bits)	1	1	1	1	1	1	1
# of ADD (32 bits)	0	1	0	0	0	1	0
# of MULT (16 bits)	1	1	0	1	1	1	1
# of MAC (16 and 32 bits)	0	0	1	0	0	0	0
Addressing unit type t_{addr}	-	-	-	-	2	-	3
# of address registers n_{dp}	-	-	-	-	8	-	6
# of index registers n_{dn}	-	-	-	-	0	-	6
# of loop registers n_{loop}	0	0	0	0	0	0	4

TABLE VII
COMPARISON BETWEEN ESTIMATED PROCESSOR CORE AREA AND LOGIC-SYNTHESIZED PROCESSOR CORE AREA.

Processor id	Logic-synthesized area [μm^2]	Estimated area [μm^2]	Error [%]
1	1123339	1116622	0.60
2	1230075	1222492	0.62
3	1126705	1128257	0.14
4	1047557	1035842	1.12
5	1122994	1134997	1.06
6	1451395	1452315	0.06
7	2131653	2143228	0.54

TABLE VIII
COMPARISON BETWEEN ESTIMATED PROCESSOR CORE DELAY AND LOGIC-SYNTHESIZED PROCESSOR CORE DELAY.

Processor id	Logic-synthesized delay [ns]	Estimated delay [ns]	Error [ns]
1	8.41	8.61	0.20
2	8.01	8.61	0.60
3	10.45	11.93	1.48
4	9.51	10.36	0.85
5	9.22	10.36	1.14
6	10.11	10.55	0.40
7	9.62	10.55	0.93

and errors of delay estimation are less than 2ns when comparing estimated area and delay with logic-synthesized processor core area and delay.

VI. CONCLUSIONS

In this paper, we proposed area and delay estimation equations for digital signal processor cores. In the future, we will establish power estimation equations for digital signal processor cores.

ACKNOWLEDGMENTS

The authors would like to thank D. Yoshizawa of Waseda University for his implementations and valuable discussions. This research is conducted as a part of the project: "Image Recognition/Description and Geographic Information System" at the Advanced Research Institute for Science and Engineering, Waseda University. This research was supported in part by Grant-in-Aid for Scientific Research Nos. 10650345 and 12750369 from the Ministry of Education, Science, Sports, and Culture of Japan.

REFERENCES

[1] H. Akaboshi and H. Yasuura, "COACH: A computer aided design tool for computer architects," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E76-A, no. 10, pp. 1760–1769, 1993.

[2] A. Alomary, T. Nakata, Y. Honma, M. Imai, and N. Hikichi, "An ASIP instruction set optimization algorithm with functional module sharing constraint," in *Proc. ICCAD-93*, pp. 526–532, 1993.

[3] N. N. Bình, M. Imai, A, Shiomi, and N. Hikichi, "A hardware/software partitioning algorithm for designing pipelined ASIPs with least gate count," in *Proc. 33rd DAC*, pp. 527–532, 1996.

[4] N. N. Bình, Masaharu Imai, and Yoshinori Takeuchi, "A performance maximization algorithm to design ASIPs under the constraint of chip area including RAM and ROM sizes," in *Proc. ASP-DAC'98*, pp. 367–372, 1998.

[5] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan-Kaufman, 1990.

[6] I.-J. Huang and A. M. Despain, "Synthesis of instruction sets for pipelined microprocessors," in *Proc. 31st DAC*, pp. 5–11, 1994.

[7] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee, *DSP Processor Fundamentals: Architectures and Features*, Berkeley Design Technology, Inc., 1994–1996.

[8] V. K. Madisetti, *Digital Signal Processors*, IEEE Press, 1995.

[9] N. Togawa, M. Yanagisawa, and T. Ohtsuki, "A hardware/software cosynthesis system for digital signal processor cores," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E82-A, no. 11, pp. 2325–2337, 1999.

[10] N. Togawa, M. Yanagisawa, and T. Ohtsuki, "A hardware/software cosynthesis system for digital signal processor cores with two types of register files," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E83-A, no. 3, pp. 442–451, 2000.