PAPER

Interacting Self-Timed Pipelines and Elementary Coupling Control Modules

Kazuhiro KOMATSU[†a)], *Student Member*, **Shuji SANNOMIYA**[†b)], **Makoto IWATA**[†,††], *Members*, **Hiroaki TERADA**[†††], *Fellow, Honorary Member*, **Suguru KAMEDA**[††], *and* **Kazuo TSUBOUCHI**[††], *Members*

SUMMARY The self-timed pipeline (STP) is one of the most promising VLSI/SoC architectures. It achieves efficient utilization of tens of billions of transistors, consumes ultra low power, and is easy-to-design because of its signal integrity and low electro-magnetic interference. These basic features of the STP have been proven by the development of self-timed data-driven multimedia processors, DDMP's. This paper proposes a novel scheme of interacting self-timed (clockless) pipelines by which the various distributed and interconnected pipelines can achieve highly functional stream processing in future giga-transistor chips. The paper also proposes a set of elementary coupling control modules that facilitate various combinations of flow-thru processing between pipelines, and then discusses the practicality of the proposed scheme through the LSI design of application modules such as a priority-based queue, a mutual interconnection network, and a pipelined sorter.

key words: *self-timed pipeline, flow-thru processing, interacting pipelines, VLSI/SoC architecture*

1. Introduction

With the continuous advancement of semiconductor technology, tens of billions of transistors are available on a single die at present. Pipeline structure is considered to be the best solution to overcome ULSI design limitation constraints through a 'divide and conquer' design. In addition, the pipeline structure also helps localize wiring over a die and thus minimizes extrinsic degradations. However, with an increase in the clocking rate, the structure suffers from the excessive power consumption and signal integrity problems associated with synchronous clock distribution. In order to solve these problems simultaneously, a basic self-timed pipeline (STP) structure has been utilized to develop a commercial data-driven multimedia processor (DDMP) [1] and a unique self-timed functional module [2].

This paper proposes an advanced scheme of interacting self-timed pipelines to fully utilize giga-transistors on a die area. The key to the proposed scheme is to allow every stage of the ordinary linear-pipeline structure to interact among several pipelines. Based on this idea, several streams of data and their pipelined processing can be flexibly deployed on the 2D plane of the vast die by virtue of the autonomous and distributive behavior of the STP. In this paper, a set of elementary coupling control modules for the interacting self-timed pipelines is proposed in order to facilitate all possible interactions between two STP stages. Then, the practicality of the proposed scheme is shown through LSI design of some example modules for dedicated functions; priority queue, mutual interconnection network, and pipelined sorter.

Manuscript received October 6, 2008.
Manuscript revised February 15, 2009.

[†]The authors are with the Graduate School of Engineering, Kochi University of Technology, Kami-shi, 782-8502 Japan.
[††]The authors are with Research Institute of Electrical Communication, Tohoku University, Sendai-shi, 980-8577 Japan.
[†††]The author is with SOFTBANK TELECOM Corp., Yokohama-shi, 221-0055 Japan.
a) E-mail: 105407v@gs.kochi-tech.ac.jp
b) E-mail: sannomiya.shuji@kochi-tech.ac.jp

DOI: 10.1587/transfun.E92.A.1642

2. Interacting Self-Timed Pipelines

Pipeline processing can be categorized into two groups, deterministic and non-deterministic, by focusing on the determinacy of the operands in the streams of data. The operands of the former are predetermined before the run of the processing, and those of the latter are not predetemined but determined at run-time. To realize the operation on the value and the order of both deterministic and non-deterministic operands, not only data processing such as calculation and comparison but also flow control such as timing adjustment among pipeline stages are necessary. The data processing and flow control among pipeline stages are referred to collectively as interaction, and the pipeline processing based on the interactions is called flow-thru processing in this paper.

In this section, the basic features of a self-timed pipeline which is preferable to realize the flow-thru processing are briefly introduced, and then possible interactions between two pipelines are discussed and classified on the basis of the interacting self-timed pipelines.

2.1 Self-Timed Pipeline

The basic structure of a self-timed pipeline (STP) is illustrated in Fig. 1. In the STP, each stage consists of a data latch

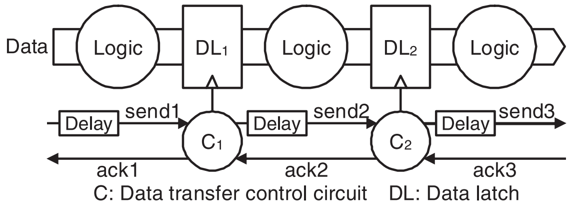


Fig. 1 Basic structure of self-timed pipeline.

(DL), a function logic (Logic), and a C element (C) which provides the coincidence function of Muller's C-element [3] to control data transfer. A piece of data, which is termed packet in the STP, is transferred locally between adjacent pipeline stages. This local data transfer is controlled based on a bundled data transfer scheme using transfer request (send) and acknowledge (ack) signals [1]. The C element is designed to assert its send signal to the next stage and its gate open signal only when the input send and ack signals are coincident. The delay element (Delay) delays the arrival of the send signal to ensure completion of the Logic.

This local transfer control of an STP makes the wiring localized and also makes data transfer on each stage autonomous. The localized wiring results in not only a lower skew, making it easier to assure timing constraints and signal integrity, but also the natural concentration of power consumption in the processing stage. Therefore, it is easy to achieve high throughput by deep pipelining.

The autonomous data transfer results in the elimination of centralized control and scheduling on the processed and transferred data, and even if the arrival time of transferred data at a stage is non-deterministic, each stage can definitely process the data according to the timing and its neighboring stages' situations. In contrast, in clock-synchronized pipelines, the non-deterministic arrival time makes it difficult to assure data transfer because it is necessary to observe all local events at each stage and their influences.

2.2 Interaction between STP's

From the concept of dataflow computation, input streams of data are operated using a pipeline processing scheme and the output streams are produced as a result of the interactions among input streams. Generally, the interactions involve not only simple dividing and merging of the streams but also other complex flows with processing, such as data exchange based on comparison, calculation of data values, and integration of a set of data. Thus, the direct pipelining of flow-thru processing is effective in achieving thorough exploitation of the parallelism inherent in the target algorithms or applications.

A previous study revealed the ability of flow-thru processing to realize flexible pipeline structures with high functionality, such as priority-based queuing module [2]. However, as far as the authors know, all possible interactions even between two pipelines have never been clarified.

(a) Static Path of Dataflow

To realize arbitrary interaction in self-timed pipelines, the dataflow between pipelines should be arbitrarily controlled while the data are processed. Topologically, an arbitrary dataflow path can be provided by one of the combinations of outflows from one pipeline to the other. Figure 2 illustrates the variation of outflows between pipelines, named A and B. In the figure, A_b and B_b denote branch point, while A_m and B_m denote merge point. In addition to the forward paths (straight or cross), a sideward path (A_b to B_b) is useful

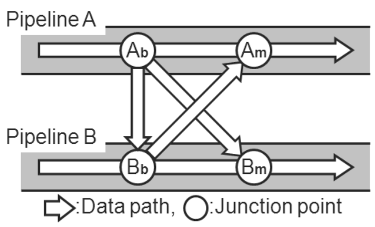


Fig. 2 Possible dataflow paths.

Table 1 Useful dataflow combination (from pipeline A to B).

Interaction name (based on static data path)		Active paths		
		to A_m	to B_m	to B_b
Divide	A_b	○	○	—
	B_b	—	—	—
Merge	A_b	—	○	—
	B_b	—	○	—
One-way	A_b	○	○	—
	B_b	—	○	—
Bi-directional	A_b	○	○	—
	B_b	○	○	—
Sideward one-way	A_b	○	—	○
	B_b	○	○	—

○: existent, —: nonexistent

to achieve the interaction with one pipeline whose dataflow is temporally dammed. All the possible combinations of dataflow paths are listed in Table 1, in which functionally identical paths are omitted to simplify the table. For instance, the Merge from A_b and B_b to A_m is symmetrically equivalent to that to B_m.

(b) Dynamic Control of Dataflow

On the static paths, operands of operation should be detected during an interaction. In the interactions in the deterministic processing, it is necessary to detect proper operands exactly among data streams. In contrast, it is important to change the number of operands according to the number of arrived data in the interactions in the non-deterministic processing, in order to avoid the throughput degradation of the flow-thru processing. To realize such interactions for both deterministic and non-deterministic processing, two kinds of interaction modes are introduced here, and they are called deterministic mode and non-deterministic mode in this paper. In the case of deterministic mode, datum of an input stream flowing in one pipeline is forced to wait for the datum on the other pipeline and then the coupled data interact with each other. In the case of non-deterministic mode, two data flowing in two pipelines interact only when they arrive at A_b and B_b within a predesignated time.

The interaction can be realized by combining the static data flow path and the dynamic mode. During an interaction, two input data coupled according to the mode may be operated, and then the destination and output priority of resultant data are decided based on some of the following information.

- Operation result of two data
- Arrival order of two data, e.g., a first-come-first-served

Table 2 Possible transfer control (from pipeline A to B).

	Transfer control
Divide	$A_b \rightarrow A_m$
	$A_b \rightarrow B_m$
Merge	$A_b \rightarrow B_m$
	$B_b \rightarrow B_m$
One-way	$A_b \rightarrow A_m$, $B_b \rightarrow B_m$
	$A_b \rightarrow B_m$
	$B_b \rightarrow B_m$
Bi-directional	$A_b \rightarrow A_m$, $B_b \rightarrow B_m$
	$B_b \rightarrow A_m$
	$A_b \rightarrow A_m$
	$A_b \rightarrow B_m$
	$B_b \rightarrow B_m$
	$A_b \rightarrow B_m$, $B_b \rightarrow A_m$
Sideward one-way	$A_b \rightarrow A_m$, $B_b \rightarrow B_m$
	$B_b \rightarrow A_m$
	$A_b \rightarrow A_m$
	$A_b \rightarrow B_b$, $B_b \rightarrow B_m$
	$A_b \rightarrow B_b$
	$B_b \rightarrow A_m$

- Dataflow rate of two pipelines
- Specific attribute of the pipeline stage, e.g., stage number

Finally, the resultant data are routed along with the topological dataflow path described in Fig. 2. The possible combinations of the dataflow paths are listed in Table 2. In an interaction, if the routes of data transfer have the same destination, conflict occurs. In the case of conflict, the data transfers are ranked based on the above information, and then each data transfer is selected and controlled exclusively.

In the interaction, a dataflow in one pipeline should be temporally dammed when the pipeline accepts inflow from the other pipeline. Even with the STP, if such transfer control is spread over multiple pipeline stages, it results in the waste of circuit area and also complex control becoming a bottleneck. That is, interaction control should be localized between corresponding stages of two pipelines. Although, fork and join controls [5] have been used to connect and control the pipelines, they are not assumed to be used within the same stage and thus the transfer control may spread over several stages. For instance, a bi-directional control should be realized by two join stages followed by two fork stages in order to realize the simultaneous transfer controls $A_b \rightarrow A_m$ and $B_b \rightarrow B_m$. In contrast, existing pipeline controls, such as a control in counterflow pipeline [6], can be used as an interaction control, however their transfer paths are limited and there are no controls providing all of paths between two pipelines. Therefore, an interaction control circuit should be newly designed, so that the flow-thru processing can be easily realized by interacting self-timed pipelines.

3. Interaction Circuit

By imposing operational conditions, such as arrival timing and specific values of data sets, in flow-thru processing, interesting interactions between STP's can be realized. However the manual design of circuits to control the coupling of

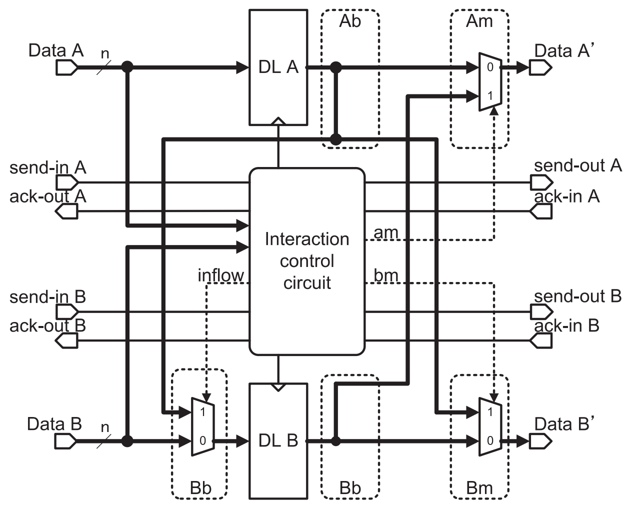


Fig. 3 Interaction stage.

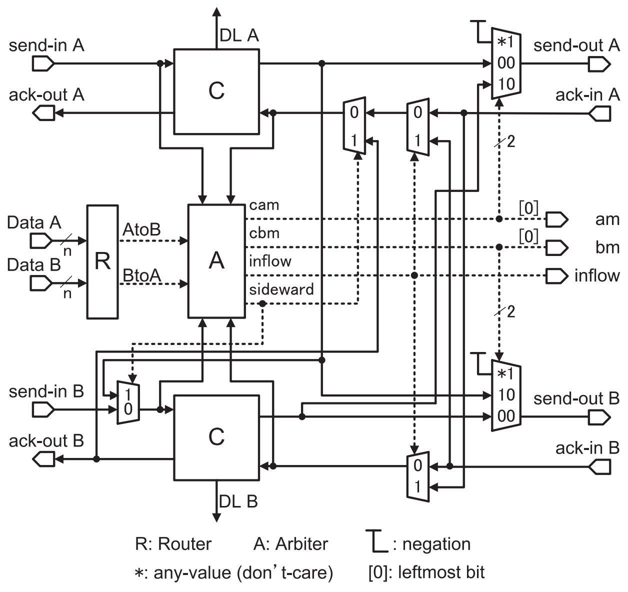


Fig. 4 Interaction control circuit.

the STP's may take a long period of time and will be impossible for future large scaled systems because assuring the timing of handshakes requires comprehensive understanding of all signals changing asynchronously during interactions. In this section, coupling control modules to control flow-thru processing are proposed, in addition, design constraints are revealed to modularize them.

3.1 Elementary Coupling Control Modules

To facilitate hazard-free handshake, the coupling stages of STP's are structured as illustrated in Fig. 3, and their interaction control circuit consists of four modules, as illustrated in Fig. 4. Each data latch is controlled by the C element (C). Since the send-out and ack-in signals should be appropriately selected along with the forward paths (straight or cross) of routed data, 5 multiplexers (MUX's) are intro-

Table 3 Arbitration for interaction (from pipeline A to B).

	Route		Selected	Arbitration			
	AtoB	BtoA	transfer control	cam	cbm	inflow	sideward
Divide	0	*	Ab → Am	00	*1	0	0
	1	*	Ab → Bm	*1	10	1	0
Merge	*	*	Ab → Bm	*1	10	0	0
			Bb → Bm	*1	00	1	0
One-way	0	*	Ab → Am, Bb → Bm	00	00	0	0
	1	*	Ab → Bm	*1	10	1	0
			Bb → Bm	*1	00	0	0
Bi-directional	0	0	Ab → Am, Bb → Bm	00	00	0	0
	0	1	Bb → Am	10	*1	0	0
			Ab → Am	00	*1	0	0
	1	0	Ab → Bm	*1	10	1	0
			Bb → Bm	*1	00	0	0
	1	1	Ab → Bm, Bb → Am	10	10	1	0
Sideward one-way	0	0	Ab → Am, Bb → Bm	00	00	0	0
	0	1	Bb → Am	10	*1	1	1
			Ab → Am	00	*1	0	0
	1	0	Ab → Bb, Bb → Bm	*1	00	0	1
	1	1	Ab → Bb	*1	*1	0	1
			Bb → Am	10	*1	1	1

duced between C elements or among C elements. Furthermore, one more MUX is inserted on the send-in B signal line to realize the handshake of the sideward path between two pipelines. Because every MUX should be controlled by its select signal, an arbiter (A) generates select signals to arbitrate two pipelines. The arbiter observes all input signal changes of two C elements and adjusts the timing of the change of data transfer route. That is, the arbiter can detect the order of the arrival timing of two input data and then arbitrate them along with the order, e.g., in a first-come-first-served manner. A router (R) operates the information depending on the requirement specification of the target application. The router decides the destination and output priority of the forwarded data based on the data (or a part of the data) of two input packets before arbitration, and outputs them to the arbiter. The outputs of the router and arbiter modules are drawn by a dashed line to make it easy to recognize the wiring. The values of the output signals of the arbiter and router modules are listed comprehensively in Table 3. With this module division, the C element can be designed independently from other modules, and a hazard free handshake can be realized by using an asynchronous circuit design method. The MUX, arbiter, and router modules can be shrunk according to the combination of data flow paths, interaction mode, and operational conditions.

(a) C element: C

Several design methods have been already studied for asynchronous circuits including self-timed circuits, and they are categorized into two types, Huffman circuit and Muller circuit [3]. The Muller circuit assumes a circuit element, called a generalized C-element. Unfortunately the generalized C-element is not supported in a usual standard cell library of LSI manufacturers and it is difficult to guarantee its stability under the deep-sub-micron process, and thus it is not widely accepted. In contrast, the Huffman circuit can be designed using standard logic gates and therefore it is adopted in our circuit.

The Huffman circuit was proposed with a procedure to derive circuit from its specification described by burst-mode machine, which is a kind of state machine. In the procedure of circuit derivation, the number of signals changing simultaneously is constrained to assure the derivation of hazard-free circuits. In the case of multiple input changes, an extended burst-mode (XBM) machine [7] is often utilized to specify the asynchronous circuit.

To satisfy the constraints of the XBM machine, a 4-phase handshake protocol between two C elements is introduced to limit the number of signals changing simultaneously. To transfer a packet from the i-th to the $(i+1)$-th stage in Fig. 1, the 4-phase handshake is performed as below.

1. C at the i-th stage (C_i) asserts its send-out signal to C_{i+1} (*send-out*$_{i+1}$), and it also asserts its gate open signal to DL_i.
2. In response to the assertion of the *send-out*$_{i+1}$, the C_{i+1} asserts *ack-out*$_{i+1}$ to the C_i.
3. The C_i receiving the *ack-out*$_{i+1}$ negates the *send-out*$_{i+1}$ and the gate open signal.
4. After the negation of the *send-out*$_{i+1}$, the C_{i+1} negates the *ack-out*$_{i+1}$. The C_{i+1} also asserts the *send-out*$_{i+2}$ only when the *send-out*$_{i+1}$ and *ack-out*$_{i+2}$ are negated i.e. they are coincident. After the negation of the *ack-out*$_{i+1}$, the C_i becomes ready to receive the *send-out*$_i$.

The same procedure is repeated between adjacent stages. The handshake based on the introduced protocol is a negative logic, i.e., the assertion and negation correspond to low and high level signals respectively. The timing chart illustrated in Fig. 5 shows that the assertion of ack signal is postponed until the next stage becomes empty, i.e., the *ack-out*$_i$ is asserted after the *send-out*$_i$ is asserted and the *ack-out*$_{i+1}$ is negated. This 4-phase handshake protocol experiences pipeline throughput degradation only when an overload situ-

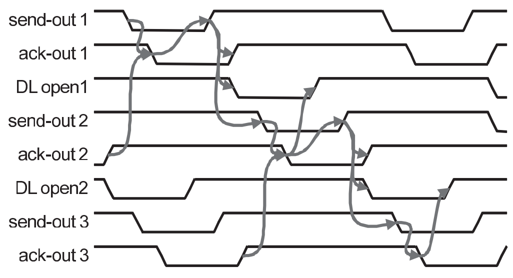


Fig. 5 Timing chart of handshake.

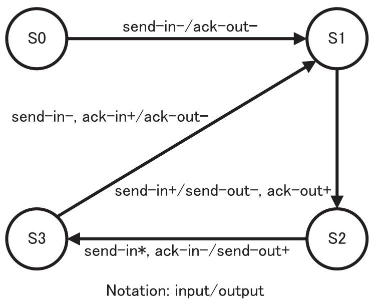

S0
S1
S2
S3
send−in−/ack−out−
send−in−, ack−in+/ack−out−
send−in+/send−out−, ack−out+
send−in*, ack−in−/send−out+
Notation: input/output

Fig. 6 XBM for C element.

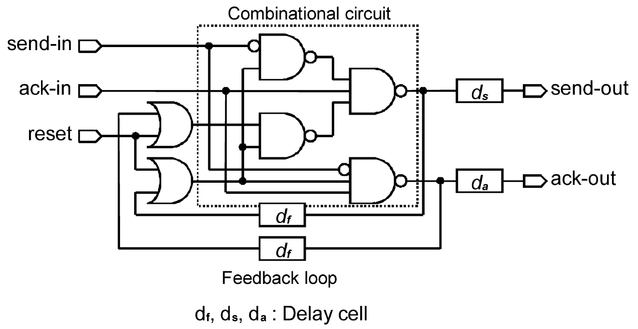


Fig. 7 Huffman circuit of C element (C).

ation occurs in the pipeline. This is because the ack signals' delays, which are usually concealed by delays in the send signals, influence the effective data transfer time in an overload situation. However, this drawback can be overcome in sophisticated systems because the overload situation can be avoided by a load balancing scheme.

The XBM machine specification for the C element is synthesized based on the introduced protocol, and its correctness is checked by evaluating all events among the XBM's. Figure 6 shows the synthesized specification. In this figure, state S0 denotes the initial state when the circuit is reset. After that, states S1, S2, S3 will be taken along with the handshake signal changes. The rising edge (+) and the falling edge (−) of the send-in, send-out, ack-in and ack-out signals are denoted by send-in+, send-in−, send-out+, send-out−, ack-in+, ack-in−, ack-out+, and ack-out−. The *directed don't-care* (send-in∗) indicates that the send-in signal might either keep its level or change exactly once.

The Huffman circuit for the illustrated XBM machine can be algorithmically derived. This requires little time if the algorithm is supported by CAD. The derived circuit is illustrated in Fig. 7, and consists of a combinational circuit to determine the next state, a feedback loop for state transition, and delay cells (d_f) to hold the current state. The required delay time of each delay cell will be discussed in the next sub-section.

(b) Arbiter module: A
As described above, arbiter (A) decides a flow path based on not only the output of the router module but also all of the input signals of the two C elements. The AtoB signal represents that data in pipeline A is transferred to pipeline B when it is '1' and it represents that data in pipeline A is transferred to pipeline A when it is '0'. Similarly, the BtoA represents the transfer direction of the data in pipeline B.

When the AtoB and BtoA signals indicate the same destination and imply conflict, the arbiter select one of the transfer control based on the arrival times of the input send-in signals. To facilitate the selected transfer control, the arbiter controls the MUXs by its output signals described in Table 3.

In the case of non-deterministic interaction, the arbiter initiates the selection of the transfer controls once one of its send-in signals falls, and it outputs control signals to the MUXs. Then, the arbiter retains its output signals until one data transfer is completed, i.e., until the corresponding ack-in rises. Finally the arbiter releases the flow path immediately. If a packet arrives on a different pipeline during the transfer of the packet arriving first, the packet on the different pipeline is treated as late one.

In the case of deterministic interaction, if either of two send-in signals falls, the arbiter instructs the MUXs to stop propagating its send-out signals to the anterior C element. The arbiter selects the transfer control after the other send-in falls, and assures the selected transfer control by keeping its output signals until ack-out A and B rise. Finally the arbiter releases the flow paths immediately.

(c) Router module: R
The router (R) is a combinational circuit that decides the destination according to the data values, and thus its implementation depends on the operational conditions specified in target applications. For instance, the router may be a comparator that compares the priority field of data.

3.2 Timing Constraints

To assure the C element remains hazard-free, the timing constraints on the proposed coupling control modules are discussed according to the fundamental constraints of Huffman circuit.

Huffman circuit is an asynchronous state machine, and provides a delay in its state feedback loop to hold the state. To assure the state transitions of Huffman circuit, the minimum interval time between input changes must be larger than $2d_{max} + d_f$ where d_{max} and d_f denote the maximum delay amount from the input to the output of the combinational circuit, and the delay amount of the feedback loop separately [3].

During a handshake with the proposed protocol, the ack-in+ and send-in− may occur simultaneously, but this simultaneous changing is not sensed as input changes because the send-in− is defined as a directed don't-care [3]. Except for this simultaneous change, the minimum interval time, which is denoted d_i, is determined by one of three input changes; from send-in− to send-in+, from send-in+ to ack-in−, and ack-in+ to ack-in−. To make the explanation simple, the structures and delays of the adjacent stages i.e. the anterior ($stage_{i+1}$) and posterior ($stage_{i-1}$) interaction stages are assumed to be the same as those of the focusing stage ($stage_i$). By tracing the input change along with the wiring between logic gates, it is revealed that the minimum interval times of those input changes are the same and defined as $2d_{min} + d_s + d_a + 3d_{mux}$. The d_{min} is the minimum delay from the input to the output of the combinational logic in Huffman circuit, and the d_{mux} is the delay of MUX. The delay time, which is denoted d_s, is for delaying the arrival of the send-out signal to guarantee the completion of logic operation, while the delay time, which is denoted d_a, is for delaying the arrival of the ack-out signal to guarantee the setup-hold time of a data latch. Consequently, the constraint of the C element is defined as:

$$2d_{min} + d_s + d_a + 3d_{mux} > 2d_{max} + d_f. \qquad (1)$$

As for the cases where different interactions are placed at $stage_{i-1}$ or $stage_{i+1}$, some delay times are replaced based on the actual circuit structure. In addition to the constraint of the C element, the timing constraint for the router (R) and arbiter (A) are also revealed. The router can be considered as a part of logic function of a stage, and thus the completion of its logic is assured by adjusting the d_s, as similar to the other pipelined logic functions. In contrast, the arbiter detects the arrival order by observing the assertion of the send-in (send-in−), and it decides the arbitration signals with the output of the router. The output of the arbiter and MUXs must be decided before the assertion of the send-out signal (send-out−) arrives. That is, the delay amount of the arbiter, denoted as d_{arb}, must be within the time from send-in− to send-out+. The input change from send-in− to send-out− experiences ack-out−, send-in+, and send-out−. According to the handshake protocol illustrated in Fig. 5, first the send-in− turns out the ack-out− at a focusing C element. Next the ack-out−, which is the ack-in− to the C element of $stage_{i-1}$, turns out send-out+, which is send-in+ for $stage_i$. After that, the send-in+ turns out send-out− at $stage_i$. That is, the signal passes through the C element three times, and the minimum amount of the time is defined as $3d_{min}+2d_s+d_a+3d_{mux}$. Within this time, the delay from the input of the arbiter to the output of the MUXs, denoted as $d_{arb} + d_{mux}$, must be adjusted. Consequently, the constraint of the arbiter module and MUX is defined as:

$$3d_{min} + 2d_s + d_a + 3d_{mux} > d_{arb} + d_{mux}. \qquad (2)$$

To satisfy the timing constraints against the timing variation which is often caused by PVT variation, the minimum amount are assumed for the delays in the left hand side of the expressions, while the maximum amount assumed in the right hand side.

The derived constraints show a time margin to reduce metastability. Because the arbiter module is designed using an asynchronous sequential circuit, the probability of metastability occurence has to be reduced for practicality. One of the solutions to the metastability is to cascade the RS flip-flops in the sequential circuit to reduce the probability of the metastability occurrence as low as practically acceptable [3]. In our circuit, two flip-flops are cascaded redundantly to reduce the probability.

The flow-thru processing can be guaranteed by adjusting the delay amounts to satisfy the constraints (1) and (2). The constraints lead to the elimination of manual timing adjustment for all combinations of delays on every circuit element, and thus makes it dramatically easy to design interacting self-timed pipeline systems. The advantage of the proposed modules is the modularity by which the C elements can be independent from each other while the constraints are satisfied.

4. Evaluation

The feasibility and validity of self-timed pipelines that are structured using the proposed interaction circuits is evaluated based on an LSI design of an application LSI core. The throughput and design term are measured based on their designs. In the self-timed pipeline, the handshake time among adjacent stages determines the throughput [packet/sec.], and it is measured based on RTL simulations with extracted delays on circuit cells and wiring. In addition, the design term is determined by the time spent by a master course student with two years experience on the CAD tools to complete the layouts of the circuits from HDL description.

Dedicated applications are chosen to examine one-way, bi-direction, and sideward one-way paths with either a deterministic or a non-deterministic interaction mode. The evaluated LSI cores are designed using a standard cell library of TSMC 180 nm CMOS 6 M, 1.8 V.

Each interaction stage and control circuit are derived by shrinking the generic circuit in Figs. 3 and 4. Based on the input and output specification listed in Table 3, the function of each module is verified against all of combinations of inputs. In addition, by using RTL circuit simulation with delays extracted from the placed and routed circuit cells and wires, the timing of each module is checked to satisfy the constraints revealed in Sect. 3.2.

4.1 Self-Timed Pipeline Queue

Self-timed pipeline queue (SPQ), proposed in our previous study [2], is composed of a folded pipeline where each pipeline stage interacts with its corresponding stage via its bypass route. The folded pipeline includes two opposite pipelines whose interactions are performed between oppo-

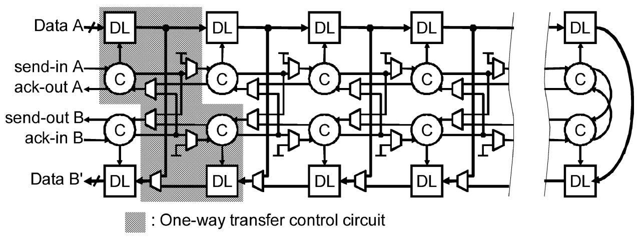


Fig. 8 Block diagram of SPQ.

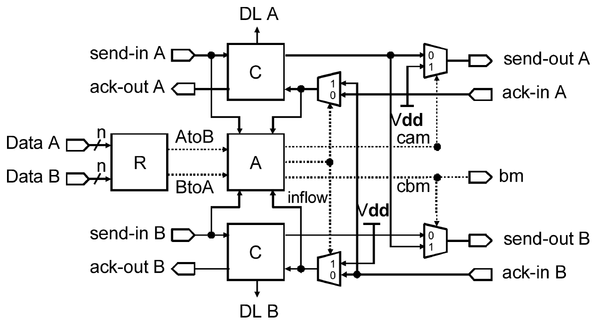


Fig. 9 One-way transfer control circuit.

site data flows differently from Fig. 2 where the flows are parallel in one direction. Figure 8 shows a block diagram of the SPQ with the bypass routes. The unique functionality of the SPQ is its prioritized variable length queuing, which enables prioritized packets to use the bypasses, even though they have no explicit scheduling functions. The SPQ is suitable for the differentiated services of QoS control [8].

The one-way dataflow path with a non-deterministic interaction mode is used to achieve a bypass route between two pipeline stages. Figure 9 shows the circuit, which is derived by shrinking the proposed circuit for SPQ. The router (R) preliminarily determines the path along with the priority bits of input data. The arbiter (A) detects whether conflict occurs or not, and the detection is realized by using data latches as shown in Fig. 10 in our circuit design. In the case of conflicts, higher priority data flowing in pipeline A is transferred to pipeline B via the bypass route. If lower priority data flows through the pipeline A, it will be transferred to the succeeding stage of A at the same time. The MUX's realize an exclusive handshake with one of the succeeding stages by stopping the propagation of the send and ack signals, and this is facilitated by pulling up the signals to V_{dd} which indicates the negated level of the send and ack signals.

The SPQ, newly designed in this paper, is composed of 128 one-way dataflow paths, i.e., it can queue up to 256 packets. The designed circuit is compared with the previous circuit [2], which was designed manually, and the results are shown in Table 4. The comparison shows that the proposed circuit can achieve 100 M packet/sec., which is equivalent to the performance of the manually designed SPQ, while the time it takes to design the SPQ is reduced to one sixth of the level for the manually designed SPQ.

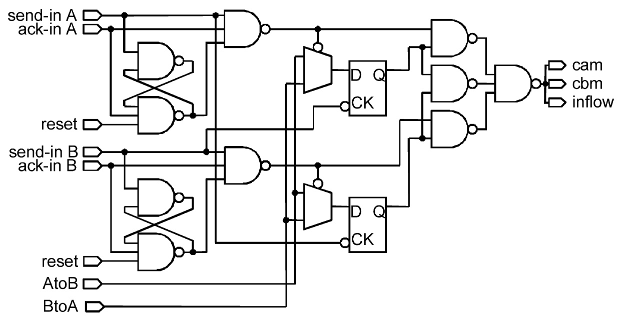


Fig. 10 Arbiter module of SPQ.

Table 4 Comparison with a manually designed circuit.

	Logic area [mm^2]	Throughput [packet/sec.]	Design term [man-month]
Proposed	1.25	100 M	0.25
Manual design	2.01	106 M	3

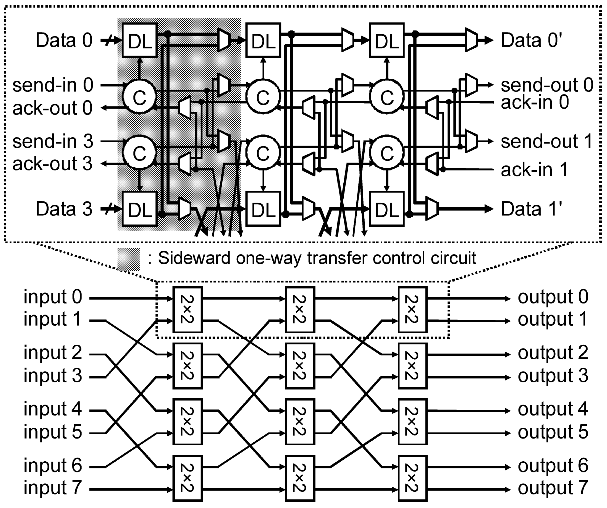


Fig. 11 Block diagram of self-timed omega network.

4.2 Omega Network

Self-timed mutual interconnection networks with a self-routing function show promise in realizing network-on-chip (NoC). In this evaluation, an omega network is chosen. The designed circuit module provides 8 ports which are derived from the typical number of processing cores. Figure 11 shows the block diagram of the designed 8 × 8 omega network with 3-bit destination address (for 8 outputs).

The 2 × 2 switching cell can be implemented by a bidirectional dataflow path with a non-deterministic interaction mode. The block diagram of the designed transfer control circuit is shown in Fig. 12. The router (R) preliminarily determines the path along with the destination address included in the input data. In the case of a conflict, i.e., if the destinations of an input data set are the same, then the first-come-first-served policy is applied. The MUXs are controlled by the arb signals to allow exclusive handshake

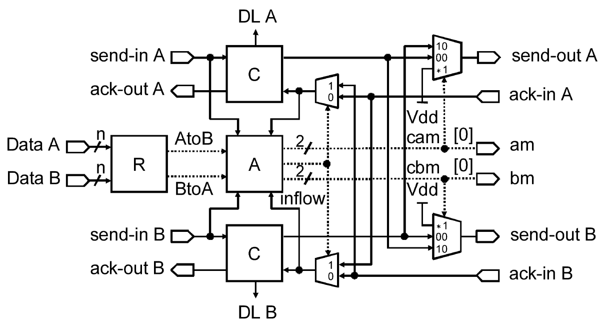


Fig. 12 Bi-directional transfer control circuit.

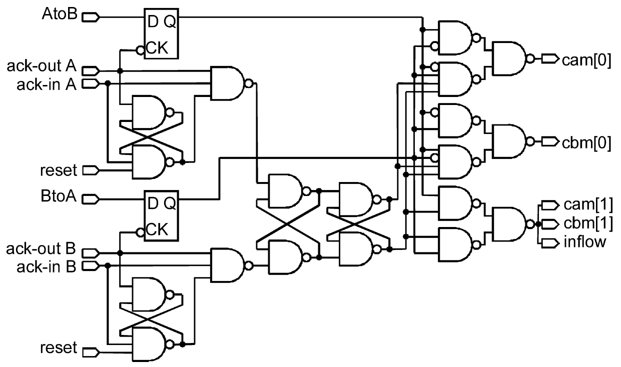


Fig. 13 Arbiter module of SPR.

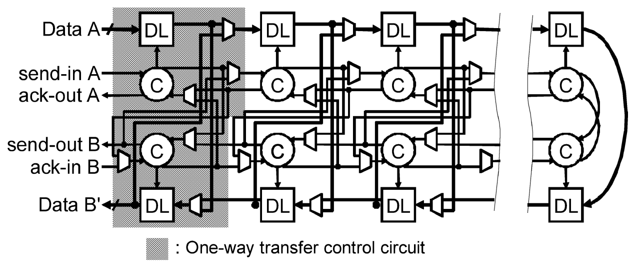


Fig. 14 Block diagram of self-timed pipeline sorter.

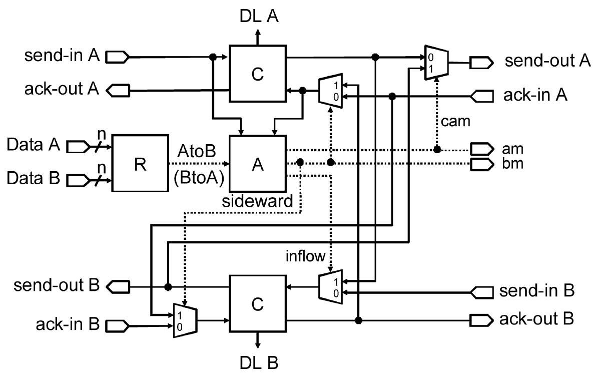


Fig. 15 Sideward one-way transfer control circuit.

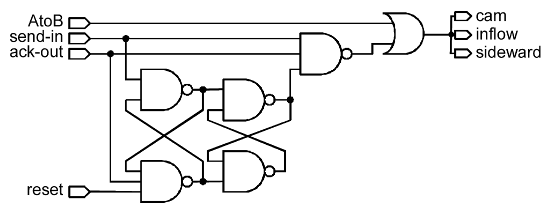


Fig. 16 Arbiter module of SPS.

to the destination stage by stopping the handshake of data which arrives a little late. Otherwise, in the case of non-conflict, the MUXs are controlled to allow simultaneous handshakes to the different succeeding stages. The arbiter module, which is illustrated in Fig. 13, detects the arrival order of a data set by using flip-flops and it determines whether simultaneous handshakes are allowed or not according to the destination (A to B and B to A signals).

The measurement shows that the designed circuit achieves 160 M packet/sec., which is enough for routing among present DDMP cores working around 140 M packet/sec. The LSI core design requires only 1/3 man-month.

4.3 Pipelined Sorter

The third application for the interacting self-timed pipelines is a pipelined sorter which provides highly parallel sorting useful for various applications, such as TCP/IP packet processing. As shown in Fig. 14, the designed sorter consists of two opposite pipelines, the input data flows in pipeline A and the ordered data is held by pipeline B. The pipeline B holds the ordered data until the sorting finishes, and thus it acts as a memory. In the corresponding stages between the two pipelines, each data in pipeline A (E_A) is compared with each data in pipeline B (E_B). If the E_B is out of order, the E_B is removed to pipeline A, and the E_A is inserted to the stage which was formerly occupied by E_B. Otherwise, the E_A steps to the next stage and is compared with the next data in pipeline B. Finally, all input data are sorted in pipeline B.

The flow-thru processing providing this comparison and swapping can be realized by a sideward one-way dataflow path with a deterministic interaction mode. The block diagram of the designed circuit is illustrated in Fig. 15. By shrinking the proposed coupling control modules, the router (R) is designed to compare the values of an input and ordered data, and the arbiter (A), which is shown in Fig. 16, is designed to detect a couple of data and to control their transfer timing by using flip-flops.

The designed pipelined sorter provides a pipeline capacity for 256 packets, which is enough for sorting in the TCP packet reassembling. The designed sorter module achieves 100 M packet/sec. This result indicates that the proposed sorter could operate at 3.2 G b/s line speed in the case of 32 bit/packet. It took only 1/6 man-month to get its layout data from the specification.

The three LSI designs of the interacting self-timed pipelines prove that highly functional circuits can be implemented easily with sufficient performance for current applications and furthermore the design period can be shortened. Although only part of the overall functionality of the interaction between pipelines is revealed, the potential of the interacting self-timed pipelines should be investigated further.

5. Conclusion

In this paper, the novel concept of interacting self-timed pipelines is proposed to distribute and interconnect various pipelines realizing highly functional stream processing on future giga-transistor chips. Moreover, the circuits of four elementary coupling control modules realizing possible interactions between STP's are proposed. The fundamental feasibility and validity of the proposed circuit for current LSI design technology are proven through the implementation of the dedicated modules.

Although only three dedicated applications with unique functionality are explored as simple examples organized by the interacting self-timed pipelines, the applicability and potentiality for general algorithms and applications should be further studied. Meanwhile, we hope to design a many-core chip as an application of the interacting self-timed pipelines.

Acknowledgement

This work was supported in part by the Dependable VLSI Research Area, Core Research for Evolutional Science and Technology (CREST), Japan Science and Technology Agency (JST). The VLSI design work was supported by VLSI Design and Education Center (VDEC) of the University of Tokyo in collaboration with Cadence Design Systems, Inc.

References

[1] H. Terada, S. Miyata, and M. Iwata, "DDMP's: Self-timed super-pipelined data-driven multimedia processors," Proc. IEEE, vol.87, no.2, pp.282–296, Feb. 1999.

[2] M. Iwata, M. Ogura, Y. Ohishi, H. Hayashi, and H. Terada, "100 MPacket/s fully self-timed priority queue: FQ," ISSCC 2004, Session 8, no.1, pp.150–151, Feb. 2004.

[3] C.J. Myers, Asynchronous Circuit Design, Wiley & Sons, 2001.

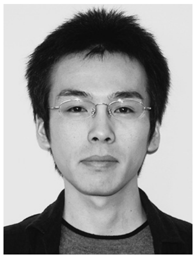
[4] J. Spars and S.B. Furber, Principles of Asynchronous Circuit Design: A Systems Perspective, Kluwer Academic Publishers, 2002.

[5] S. Furber, "Computing without clocks: Micropipelining the ARM processor," Proc. Asynchronous Digital Circuit Design, Workshops Comput., pp.211–262, 1995.

[6] R.F. Sproull, I.E. Sutherland, and C.E. Molnar, "The counterflow pipeline processor architecture," IEEE Des. Test Comput., vol.11, no.3, pp.48–59, July 1994.

[7] M.B. Josephs, S.M. Nowick, and C.H. Van Berkel, "Modeling and design of asynchronous circuits," Proc. IEEE, vol.87, no.2, pp.234–242, Feb. 1999.

[8] H. Hayashi, M. Iwata, H. Terada, and K. Shimamura, "A priority-based queueing scheme using self-timed pipeline," IEICE Trans. Commun. (Japanese Edition), vol.J87-B, no.8, pp.1063–1075, Aug. 2004.



Kazuhiro Komatsu received a B.E. degree in Department of Information Systems Engineering, Kochi University of Technology, Japan, in 2006. He is currently a master course student of the same university. His research interests are low-power pipeline architecture and its VLSI design.

Shuji Sannomiya received the B.E. and M.E. degrees in Information Systems Engineering from Kochi University of Technology, Kochi, Japan, 2002 and 2004, respectively. Now he is a research associate of Department of Information Systems Engineering, Kochi University of Technology. Currently he has interests in embedded processor architecture and LSI design.

Makoto Iwata received B.E. and M.E. degrees in Electronic Engineering from Osaka University, Osaka, Japan, in 1986 and 1988, respectively. He received a Dr.Eng. degree in Information Systems Engineering from Osaka University in 1997. Now he is a professor of Department of Information Systems Engineering, Kochi University of Technology, and a visiting professor of Research Institute of Electrical Communication, Tohoku University. His research interests are the software, architectures, and applications of novel massively parallel processing systems.

Hiroaki Terada was born in Hiroshima, Japan, on July 23, 1933. He received the B.E. degree in electrical engineering from Ehime University, Ehime, Japan, in 1956 and the M.E. and Ph.D. degrees in electrical communications from Osaka University, Osaka, Japan, in 1958 and 1964, respectively. He joined the Department of Electronic Engineering, Faculty of Engineering, Osaka University, in 1961, as a Research Associate (Assistant) and became an Assistant Professor in 1965. Specializing in control and storage technologies in electronic switching systems, he was promoted to Associate Professor of the same department in 1966. From 1976 to 1997, he served as Professor at Osaka University and was responsible for research in digital systems and education in digital systems and circuits. He also served as the Director of the Computation Center, Osaka University, from April 1993 to March 1995. Since 1997, he has been a Professor from 1997 to 2005 and the Vice-President from 2001 to 2005 at Kochi University of Technology, Kochi, Japan. He was a Visiting Professor at the University of Essex, U.K., in 1977, and a Visiting Scholar at the Centre National d 'Etudes des Telecommunications, Lannion, France, from Autumn 1977 to Spring 1978. He has also been the President of Information and Communication Laboratories of Japan Telecom Corporation since December 1997 to March 2006. Dr. Terada served as the Vice-President of the Institute of Electronics, Information and Communication Engineers of Japan (IEICE) from 1994 to 1996 and as the Kansai Regional Chairman of the Information Processing Society of Japan (IPSJ) from 1995 to 1997. He received the IEICE Achievement Award, the IEICE Kobayashi Memorial Achievement Award in 1989, the Ericsson Telecommunication Award in 2001 and the IEICE Distinguished Achievement and Contributions Award in 2003. He holds Professor Emeritus chair of Osaka University and Kochi University of Technology and currently serving as an Advisor to the Research Div. (Laboratory), SOFTBANK TELECOM Corp. His current research areas include diagrammatic data-driven language, data-driven architecture, and the VLSI-oriented implementation of data-driven architecture.

Suguru Kameda was born in Fukushima, Japan, on October 25, 1974. He received the B.S., M.S. and Ph.D. degrees in Electronics Engineering from Tohoku University, Sendai, Japan in 1997, 1999 and 2001, respectively. He is currently the Assistant Professor of the Research Institute of Electrical Communication, Tohoku University. His current interests are in heterogeneous wireless network, CDMA and OFDMA technology for mobile broadband communication systems, and RF signal processing integration circuits. He received the TELECOM System Technology Award for Student in 2001. He is a member of the IEEE.

Kazuo Tsubouchi was born in Kyoto, Japan, on February 6, 1947. He received the B.S., M.S. and Ph.D. degrees in Electronics Engineering from Nagoya University, Nagoya, Japan, in 1969, 1971 and 1974, respectively. Since 1974, he has been with the Research Institute of Electrical Communication, Tohoku University, Sendai, Japan. In 1982, he spent at Purdue University as a Visiting Associate Professor. He is currently the Professor of Wireless Info Tech Division and the director of Research Center for 21st-Century Information Technology (IT-21 Center). His current interests are highly reliable GHz-band wireless communication system, GHz-band surface and bulk acoustic wave devices and materials, low-power RF signal processing integration circuits, high speed Si CMOS circuit/device/process technology, and system in package (SiP) technology. Prof. Tsubouchi received the Hattori-Hoko Award in 1983, the 26th Ichimura Award in 1994, the TELECOM System Technology Award in 1996, the 22nd Inoue Harushige Award in 1997, the 2005 Achievement Award from IEICE, and "Minister of Education, Culture, Sports, Science and Technology Award" in the Award for Persons of Merit in Industry-Academia-Government Collaboration in FY2007. He is a member of the IEEE, the Physical Society of Japan, the Japan Society of Applied Physics, and the Institute of Electrical Engineers of Japan.