Pipelined Scheduling for Dynamically Reconfigurable FPGAs

HARASHIMA Katsumi MINAMI Yuuki KUTSUWA Toshiro

Faculty of Engineering, Osaka Institute of Technology
5-16-1, Ohmiya, Ashahi-ku, Osaka, 535-8585 Japan
Tel: +81-6-6954-4305, Fax: +81-6-6957-2136
E-mail: minami@cs.elc.oit.ac.jp

Abstract: In order to satisfy the requirement for various applications in an electronic device, many dynamically reconfigurable systems such as FPGAs have been used recently. This paper presents a pipelined scheduling for dynamically reconfigurable systems based on FPGAs. For reconfigurable systems conventional schedulings have reduced processing time by minimizing the number of reconfigurations. However, they are not effective enough for applications including many iterative processes such as digital signal processing. Our approach has been able to increase throughput of iterative applications on dynamically reconfigurable systems by using pipelined scheduling.

1. Introduction

Increasing the requirement for various applications, one needs to design various ASICs, so that many dynamically reconfigurable systems based on FPGA architecture are used[1]-[4]. These systems can use FPGAs effectively because of the dynamic reconfiguration of rewriting CLBs(Control Logic Block) in FPGAs executing of given tasks. For reconfigurable systems conventional schedulings have reduced processing time by minimizing the number of reconfigurations. However, they are not effective enough for applications including many iterative processes such as digital signal processing due to the difficulty of applying pipelined data processing for reconfigurations.

This paper proposes a pipelined scheduling for dynamically reconfigurable systems composed of three FPGAs. Our approach has been able to increase throughput of iterative applications on dynamically reconfigurable systems by using pipelined scheduling. It realizes pipeling by introducing constraints of task assignment and reconfiguration that it assigns start and end tasks into particular control steps and/or FPGAs, and do not assign any tasks into start and end control steps in particular FPGAs for a given Task Flow Graph(TFG). A pipelined schedule can execute a sequence of a TFG efficiently and use FPGAs effectively.

We have confirmed the effect of our approach by experimental results.

2. Scheduling Constraints

The proposed approach schedules under the constrains as follows;

1. A reconfigurable system takes three FPGAs(Fig.1).

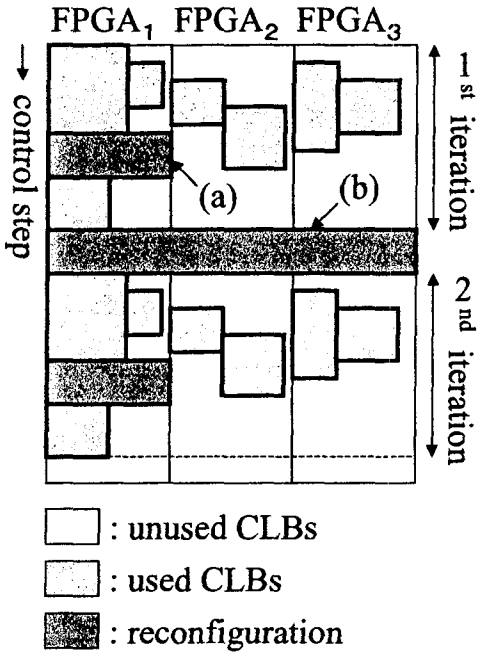

Figure 1. Target System.

2. A reconfiguration time is equal to the length of one control step(Fig.1(a)).
3. The performance and n_{CLB}, is the number of CLBs, of every FPGAs are equal.
4. An input for our approach is a TFG as shown in Fig.2. The nodes and edges represent tasks and task dependencies respectively. The *ith* node is given n_i and t_i as shown in Fig.2. n_i and t_i are n_{CLB} and execution time of task T_i individually.
5. After all tasks have finished, every FPGAs are reconfigured(Fig.1(b)). We call this a implicit reconfiguration.

Figure 3 shows two schedules for the TFG in Fig.2 where each n_{CLB} of three FPGAs is equal to 100. A schedule in Fig.3(a) can not realize a pipelining owing to the same start time of task 0 and 1(Fig.4(a)). On the other hand, a schedule in Fig.3(b) can realize a pipelining because our scheduling puts off the start time of task 1 and changes FPGAs assigning every tasks with respect to each iteration in turn(Fig.4(b)). We call this pipelining RP(rotational pipelining).

In order to obtain a pipelined schedule executing a given TFG repeatedly, the proposed scheduling introduces the restriction as shown in Fig.5(a). All tasks and reconfigurations are only assigned to assignable steps.

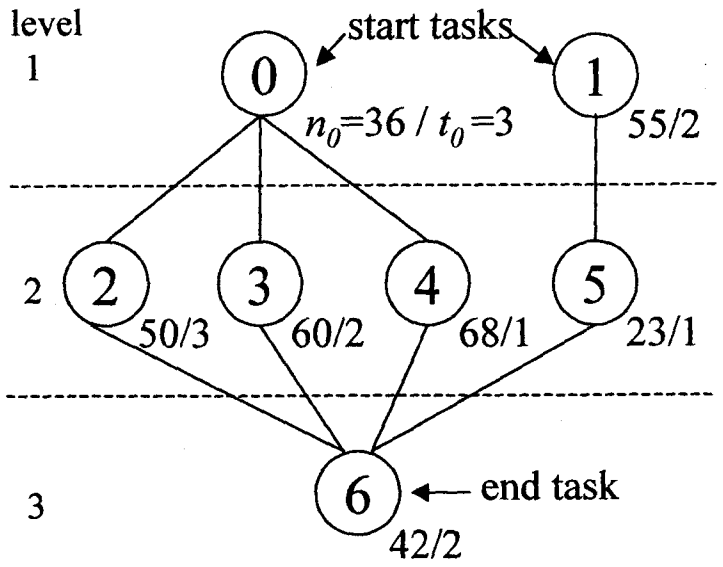

Figure 2. A Task Flow Graph.

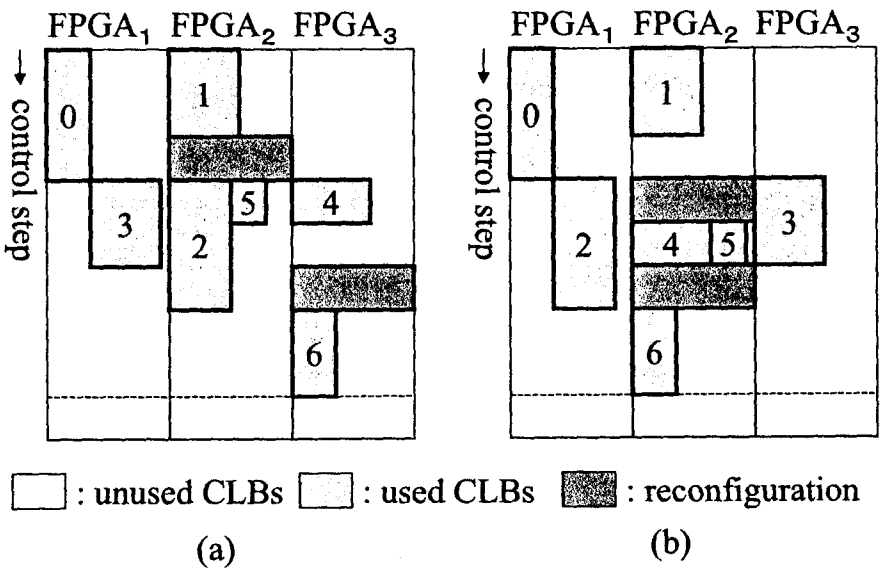

Figure 3. Schedules.

This restriction is effective for TFGs of which the ımber of start and end tasks are respectively two and ıe, and t_{diff} is not zero. Therefore, our approach takes ıly such TFGs.

3. Scheduling Algorithm

ır approach consists of the phases as follows;

Give levels to nodes of a TFG as shown in Fig.2 by ASAP Scheduling[5].

Assign the long and short tasks in the first level into $FPGA_1$(the top steps of A in Fig.5(a)) and $FPGA_2$($B1$ in Fig.5(a)) individually. As the result, t_{diff} steps following the steps assigned the short task in $FPGA_2$ are not used. Moreover, top t_{diff} and one(t_{rc}) steps in $FPGA_3$ are not used.

3. In turn from second to $final - 1$ levels, assign tasks to FPGAs as soon as possible so as to satisfy the constraints of data dependencies, n_{CLB} and scheduling described in Section 2. In each level, first assign a task T of which path length from T to the end task is largest and which has most immediate successors.
4. Assign the end task into $FPGA_2$.
5. Since 2nd iteration, realize a pipelined schedule by assigning every tasks into other FPGAs with RP(Fig.5(b)).

The purpose of our approach is to minimize the processing time of a given TFG and the throughput of an obtained pipelined schedule. In order to achieve this purpose, our scheduling takes a reconfiguration if and only if no task is able to be assigned into each FPGAs.

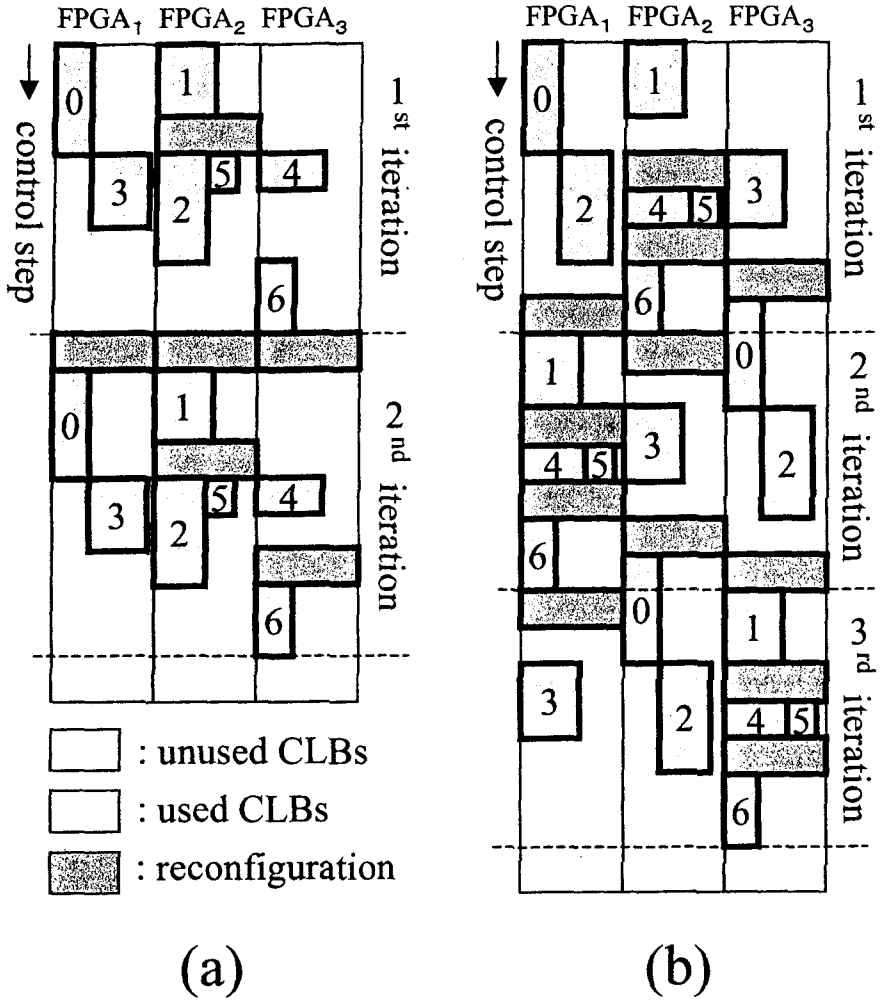

Figure 4. Pipelining.

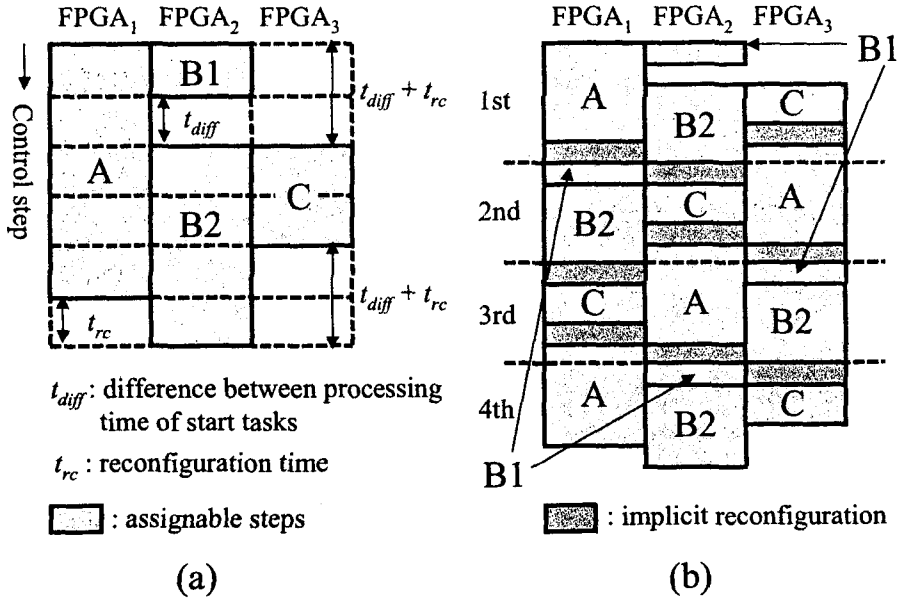

Figure 5. Constrants for Pipelining.

4. Experiments

The effect of our approach has been confirmed by experiments. The experiments used TFGs generated randomly. Table 1 shows that throughput of schedules by OURS(Our Scheduling) are only 2.5% longer than COMP(Comprehensive Search that is not pipelined). This results have indicated that our method can obtain near optimal solutions. Our scheduling results for No.10 and 14 have gotten poorer qualities than the others on the ground that their smaller t_{diff} than the others' are not effect for pipelining.

Table 2 shows that computational time of OURS is very short compared with COMP. Even if the number of nodes increases, our method can find solutions in comparatively realistic time.

5. Conclusion

In this paper, we have proposed the pipeline scheduling method for dynamically reconfigurable systems based on FPGA architecture. Our scheduling have realized pipelining by introducing constraints of scheduling. As experimental results, our method has obtained near optimal solution efficiently.

Our future works are to relax the constraints of scheduling and to minimize the number of reconfigurations.

Table 1. Experimental results of 12 nodes.

No	OURS[step]	COMP[step]	RATE[%]
1	20	20	100.0
2	21	21	100.0
3	24	24	100.0
4	15	14	107.1
5	22	22	100.0
6	35	35	100.0
7	26	26	100.0
8	26	26	100.0
9	23	22	104.5
10	22	19	115.8
11	33	30	110.0
12	23	23	100.0
13	25	25	100.0
14	18	16	112.5
15	22	22	100.0
16	18	18	100.0
17	21	21	100.0
18	16	16	100.0
19	25	25	100.0
20	17	17	100.0
Average	22.6	22.1	102.5

OURS:Our Scheduling
COMP:Comprehensive Search
RATE:OURS/COMP

Table 2. Computational time.

14 nodes.

No	OURS[step]	time[sec]	COMP[step]	time[sec]
1	27	<0.001	26	19.198
2	20	<0.001	17	8.432
3	23	<0.001	23	13.870
4	23	<0.001	22	23.614

16 nodes.

No	OURS[step]	time[sec]	COMP[step]	time[sec]
1	26	<0.001	25	291.249
2	23	<0.001	23	142.575
3	30	<0.001	29	309.805

References

[1] Toshinori Sueyoshi, Masahiro Iida, "Configurable and Reconfigurable Computing for Digital Signal Processing," IEICE TRANS. FUNDAMENTALS, VOL.E85-A, NO.3, pp.591-599, MARCH 2002.

[2] Takashi Ishitobi, Nozomu Togawa, Masao Yanagisawa, Tatsuo Ohtsuki, "A Scheduing Algorithm for a Dynamic Reconfigurable System Based on Multiple FPGAs," TECHNICAL REPORT OF IEICE, VLD2000-115, pp.33-40, 2001-1.

[3] Takashi Hakiri, Nozomu Togawa, Masao Yanagisawa, Tatsuo Ohtsuki, "A Dynamically Reconfigurable System Based on FPGAs and Its Application to a Data Encryption Algorithm," TECHNICAL REPORT OF IEICE, VLD99-109, pp.7-14, 2000-3.

[4] Junya Yamada, Toru Abe, Susumu Horiguti, "A Hardware System of Self-Reconfigurable 2D-Mesh Multiprocessor," TECHNICAL REPORT OF IEICE, VLD98-99, pp.1-8, 1998-12.

[5] Daniel Gajski, Nikil Dutt, Allen Wu, Steve Lin, "HIGH-LEVEL SYSNTHESIS Introduction to Chip and System Design," Kluwer Academic Publishers, 1992.