PAPER

Performance-Driven Architectural Synthesis for Distributed Register-File Microarchitecture with Inter-Island Delay

Juinn-Dar HUANG[†], ***Member***, **Chia-I CHEN**[†a)], **Wan-Ling HSU**[†], **Yen-Ting LIN**[†], ***and*** **Jing-Yang JOU**[†], ***Nonmembers***

SUMMARY In deep-submicron era, wire delay is becoming a bottleneck while pursuing higher system clock speed. Several distributed register (DR) architectures are proposed to cope with this problem by keeping most wires local. In this article, we propose the distributed register-file microarchitecture with inter-island delay (DRFM-IID). Though DRFM-IID is also one of the DR-based architectures, it is considered more practical than the previously proposed DRFM, in terms of delay model. With such delay consideration, the synthesis task is inherently more complicated than the one without inter-island delay concern since uncertain interconnect latency is very likely to seriously impact on the whole system performance. Therefore we also develop a performance-driven architectural synthesis framework targeting DRFM-IID. Several factors for evaluating the quality of results, such as number of inter-island transfers, timing-criticality of transfer, and resource utilization balancing, are adopted as the guidance while performing architectural synthesis for better optimization outcomes. The experimental results show that the latency and the number of inter-cluster transfers can be reduced by 26.9% and 37.5% on average; and the latter is commonly regarded as an indicator for power consumption of on-chip communication.

key words: *Behavioral synthesis, distributed register-file, performance optimization, low-power, resource binding, scheduling*

1. Introduction

As technology advances into the deep-submicron (DSM) era, interconnects are becoming one of the most crucial issues for electronic circuit and system designs. System performance, power, and area are all greatly affected by interconnects, especially for global ones [1]–[3]. It is reported that interconnects are responsible for over 50% of the overall dynamic power for a microprocessor in 130 nm technology [4]. Previous studies also show that interconnects are overwhelmingly dominating the total area and power in FPGA applications [5]–[7].

There have been several approaches proposed in the past to deal with the timing-critical issue arisen from long interconnects. Globally-asynchronous locally-synchronous (GALS) design styles adopt handshaking protocols for communication over long interconnects [8], [9]. In a synchronous latency-insensitive system (LIS), special pipelining elements, named relay stations, are inserted to break a long interconnect into shorter wire segments for sustaining higher operating clock frequency [10]–[13]. Furthermore, several types of distributed register (DR) architectures, in which the whole system is divided into several logic clusters, are also broadly studied [14]–[27]. In general, all DR-based architectures try to keep most interconnects local within a cluster and thus minimize the number of long inter-cluster interconnects for better area and performance outcome.

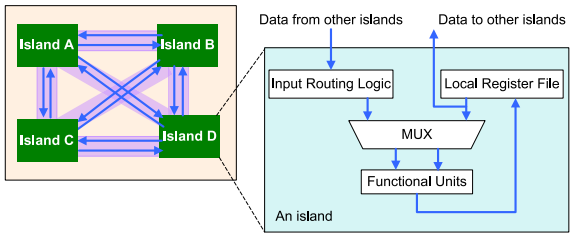


Fig. 1 (a) The DRFM architecture, and (b) the island architecture in DRFM.

The distributed register-file microarchitecture (DRFM) is one of the DR-based architectures and is recently proposed in [14], [15]. As shown in Fig. 1, a DRFM is composed of multiple islands and each of them has its own register-file, functional units (FUs), and data-routing logic. DRFM is particularly adequate for platforms with a rich set of distributed memory blocks, e.g., modern FPGAs. While utilizing DRFM, one should be aware that the way of how to map operations of a target system into islands can have a significant impact on the final outcome in terms of area and performance.

In DRFM, the notion of *inter-island connections* (IICs) is presented to better estimate the actual cost of physical global interconnects. Hence, the number of IICs is usually considered as a metric for quality of result (QoR) at early design phases [14], [15]. Meanwhile, unlike IICs, *inter-island transfers* (IITs) are the actual data transfers taking place. Multiple IITs can share one IIC as long as they have the same source-destination island pair as well as different arrival times. The number of IITs is commonly used for on-chip communication power estimation. However, during synthesis, it is not always possible to reduce both IICs and IITs at the same time; that is, there is a tradeoff between area (IIC) and power (IIT) minimization.

Inter-island delay is ignored in the original DRFM; hence its delay model appears over-simplified. To be a

Manuscript received July 1, 2011.
Manuscript revised October 1, 2011.

[†]The authors are with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.
a) E-mail: cichen.ee94g@nctu.edu.tw

DOI: 10.1587/transfun.E95.A.559

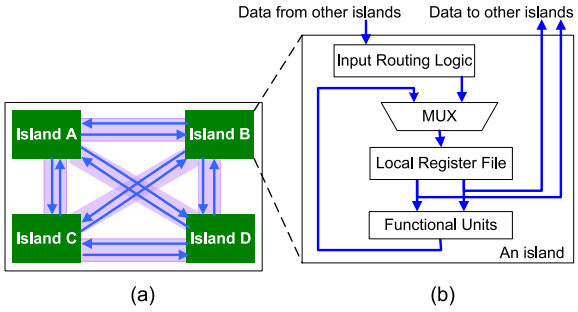


Fig. 2 (a) The DRFM-IID architecture, and (b) the island architecture in DRFM-IID.

bit more practical, here we propose a new alternative — distributed register-file microarchitecture with inter-island delay (DRFM-IID). Shown in Fig. 2(a), as one of the DR-based architecture family, DRFM-IID is also composed of multiple islands and the structure of an island is depicted in Fig. 2(b). Within an island, inputs of a local FU solely come from its local register-file (RF), whereas data from other islands (i.e., inter-island transfers) should be parked in its local RF first and become available only from the next control step (cstep). That is, an IIT takes one whole cstep for data delivery. This new interconnect delay model makes the synthesis task more complicated than the one targeting the original DRFM without inter-island delay concern since uncertain interconnect latency is very likely to seriously impact on the whole system performance. Therefore, we also develop a performance-driven architectural synthesis framework targeting DRFM-IID. Unlike previous synthesis works targeting DRFM, which focus on minimizing the number of IICs (area) [14]–[17], the proposed approach is to optimize the performance (latency in cycle count). There are two major steps in the proposed algorithm: island assignment and iterative latency minimization. At first, island assignment is responsible for properly binding operations into islands. Then, iterative latency minimization is employed to validate and further improve outcomes. The experimental results show that the proposed approach can produce synthesis results with higher performance and lower power consumption than the existing art.

The rest of this article is organized as follows. The related works and the proposed microarchitecture DRFM-IID are described in Sect. 2. Section 3 presents our key observations and motivations while Sect. 4 details the proposed synthesis algorithm. The experimental results and analyses are then given in Sect. 5, followed by the conclusions in Sect. 6.

2. Distributed Register-File Microarchitecture with Inter-Island Delay (DRFM-IID)

2.1 Related Works of DR-Based Architectures

Synthesis flows targeting DR-based architectures can be classified according to the interconnect delay model they adopt. The synthesis work is relatively easier with zero inter-cluster delay; nevertheless, this delay model appears over-simplified [14]–[17]. On the other hand, the synthesis flow considering inter-cluster delay takes a step toward reality. Obviously, with such an interconnect delay model, the synthesis task is inherently more complicated.

Regular distributed register (RDR), which is one of the DR-based architectures, and its synthesis framework are first proposed in [20]. Several enhanced synthesis frameworks [21]–[24] are also revealed later. In addition, there are two variants of the original RDR, RDR-Pipe [25] and RDR-GRS [26], [27], which focus on minimizing required interconnect resources. However, all the previously proposed synthesis frameworks targeting the RDR-based architecture do not take embedded on-chip memory or register-file into consideration.

2.2 DRFM-IID

As a member of the DR-based architecture family, DRFM-IID is composed of several clusters (islands) and each island has its own local RF, FUs and data-routing logic, as shown in Fig. 2. The fundamental difference between DRFM and DRFM-IID is the way they handle inter-island delay — DRFM-IID isolates inter-island communication delay from intra-island computation delay while DRFM does not. Be more precise, inputs of an FU can come from other islands in DRFM but always come from the local RF in DRFM-IID. That is, an entire cycle is allocated for inter-island data delivery in DRFM-IID, which makes timing convergence much easier.

In DRFM-IID, to properly model inter-island delay, a special type of node with a square shape, named a *conveyer*, is added to every IIT destination in the bound DFG. For example, Figs. 3(a) and 3(b) illustrate the same DFG mapped onto DRFM and DRFM-IID, respectively. A closed shaded region indicates that all nodes inside are bound to the same island. For $IIT_{4,2}$ (the IIT from $node_4$ to $node_2$) in Fig. 3(b), the requested data item is first sent to island I_A, then parked in the local RF at $cstep_2$. Recall that every IIT needs a complete cycle for data delivery so that the data item cannot be used in I_A until $cstep_3$. As a result, a conveyer node b is added into I_A at $cstep_2$ to reflect this fact. Furthermore, the DFG should also be rescheduled accordingly to strictly preserve data dependency. As a consequence, the latency is very likely to be stretched in DRFM-IID as demonstrated in Figs. 3(a) and (b). Note that conveyers are conceptually inserted into DFGs to preserve data dependency and not inserted as actual hardware elements into a DRFM-IID platform. A conveyer in a bound DFG indicates that a forwarded data item is received and stored in the local RF at that cstep, which costs no extra hardware resources.

Be aware the difference between the number of inter-island connections (IICs) and the number of inter-island transfers (IITs) — the former is usually less than the latter due to resource sharing. For example, in Fig. 3(a), those two IITs merely need one IIC (i.e., from I_B to I_A). As previously mentioned, in DRFM, the number of IICs has been proven to be an appropriate metric to evaluate the quality of

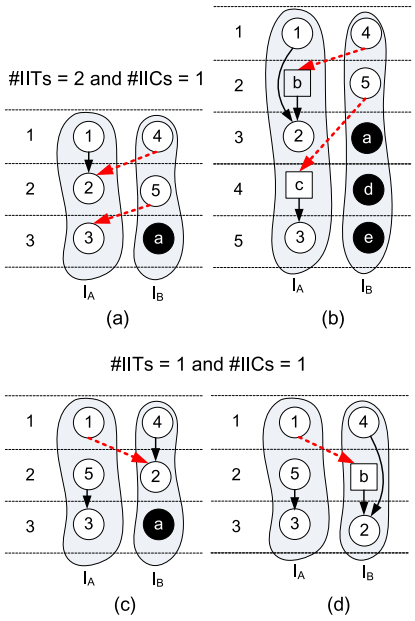


Fig. 3 (a), (b) A bound DFG with #IITs = 2 and #IICs = 1 targeting DRFM and DRFM-IID; and (c), (d) another bound DFG with #IITs = 1 and #IICs = 1 targeting DRFM and DRFM-IID.

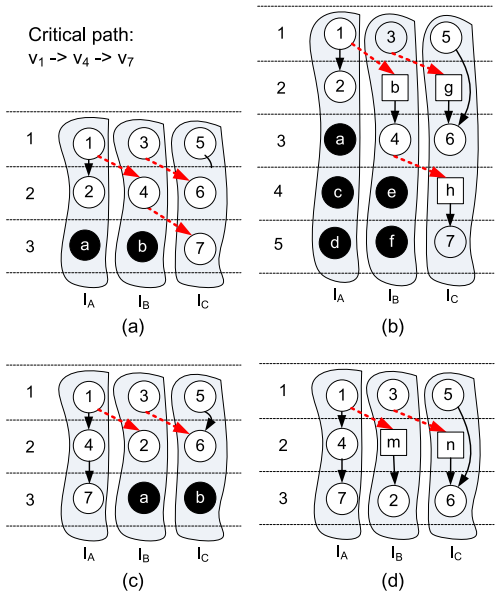


Fig. 4 (a), (b) A bound DFG w/ IITs lying on the critical path targeting DRFM and DRFM-IID; and (c), (d) another bound DFG w/o IITs lying on critical paths targeting DRFM and DRFM-IID.

result in terms of both cycle time and area at early design phases [14], [15]. Hence, if minimizing global interconnect resource is the primary goal, the main concern of synthesis flows targeting DRFM is to reduce the number of IICs instead of IITs. The previous works [14]–[17] have already done a fairly good job in terms of IIC reduction. However, things change a lot in DRFM-IID since latency may increase terribly while taking inter-cluster delay into account. That is, the number of IICs is no longer the only evaluation metric of QoR in DRFM-IID synthesis. Hence, in this work, we develop a performance-driven architectural synthesis framework targeting DRFM-IID.

3. Motivations

It is observed that the number of IITs also affects system latency in DRFM-IID. Figure 3(a) and Fig. 3(b) show the scheduling results of a bound DFG in DRFM and DRFM-IID, respectively. Similarly, Fig. 3(c) and Fig. 3(d) show the results of another bound DFG. Note that these two binding solutions have the same latency and IIC count but different IIT counts in DRFM. Those black nodes inside DFGs in Fig. 3 are named *bubbles* and are used to indicate unused (idle) time slots. Since each IIT needs a conveyer in its destination island, a bound DFG with more IITs tends to have longer latency in DRFM-IID. For instance, with additional inter-cluster delay, the binding solution in Fig. 3(b) is obviously better than that in Fig. 3(d). Consequently, besides the number of IICs (wiring resource), the number of IITs (system latency) is definitely another key metric for QoR evaluation in DRFM-IID synthesis.

In the previous example, it shows that a bound DFG with more IITs tends to have longer latency. However, to be more specific, not every IIT can effectively affect system latency in DRFM-IID. In fact, only those lying on critical paths can. Figure 4 illustrates another example. DFGs in Fig. 4(c) and Fig. 4(d) have identical latency. It is because both $IIT_{1,2}$ and $IIT_{3,6}$ do not lie on any critical paths and thus latency is not increased even after conveyer insertion in DRFM-IID. Instead, the DFG in Fig. 4(b) has longer latency than that in Fig. 4(a). The reason is that both $IIT_{1,4}$ and $IIT_{4,7}$ lie on the critical path and thus the overall latency is lengthened due to mandatory conveyer insertion in DRFM-IID. Therefore, in addition to the number of IITs, a performance-driven synthesizer for DRFM-IID should also take *timing-criticality* of IIT into account for better outcomes.

Meanwhile, the utilization of an island I, $U(I)$, is defined as (1):

$$U(I) = \frac{\#\text{ nodes in this island } + \#\text{ conveyers in this island}}{\#\text{ total csteps}} \quad (1)$$

Low island utilization implies there are many bubbles (unused time slots) in that island. In a scheduled and bound DFG, if the utilization distributes unevenly among islands, the overall system latency is very likely to be dominated by those crowded islands. For example, when a highly-utilized island, like I_A in Fig. 5(a), comes along with many incoming IITs from other islands, the overall latency is likely to be stretched after conveyer insertion, as shown in Fig. 5(b). Note that the number of in-edges of an island I represents the number of IITs ending at I. For instance, the number of in-edges of I_A in Fig. 5(a) is two (i.e., $IIT_{5,2}$ and $IIT_{9,3}$). On the contrary, the DFG in Fig. 5(c) has the same IIT count as the one in Fig. 5(a), but has no IIT ending at the highly-utilized island I_C. After considering inter-cluster delay, the resultant DFG in Fig. 5(d) has shorter latency than the one in Fig. 5(b). As a result, it is usually not a good idea to have too

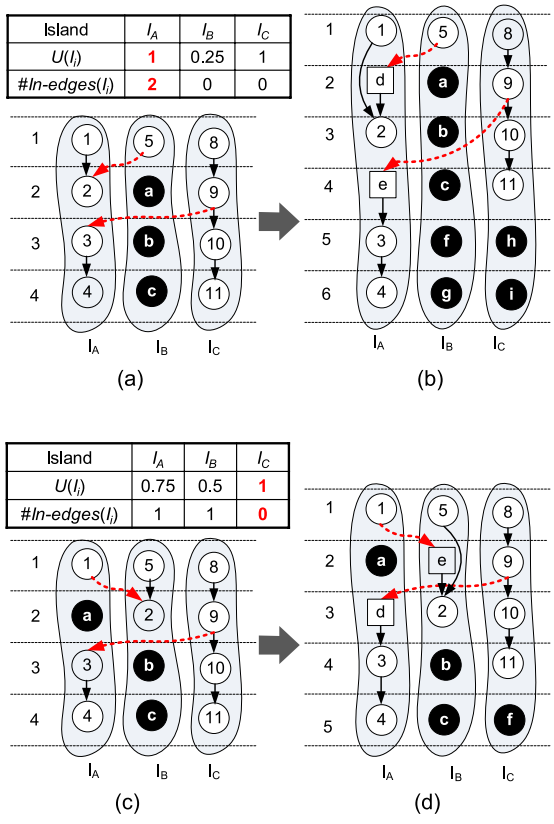


Fig. 5 (a), (b) The DFG targeting DRFM and DRFM-IID with unbalanced island utilization; and (c), (d) the DFG targeting DRFM and DRFM-IID with relatively balanced island utilization.

many in-edges incident to a highly-utilized island. That is, keeping island utilization distribution more even as well as reducing the number of in-edges for heavily-utilized islands can potentially lead to better system performance.

In summary, due to additional consideration of inter-cluster delay, architectural synthesis targeting DRFM-IID becomes even more complicated. Instead of IIC count, a performance-driven and power-aware synthesis framework should also take IIT count, which is also an indicator for on-chip communication power consumption [4], timing-criticality of IIT, and island utilization into account for better outcomes.

4. Proposed Synthesis Algorithm for DRFM-IID

As in [14], [15], it is commonly assumed that every operation can be carried out at arbitrary island. Then the problem formulation of this work can be described as: *Given a DFG and a resource constraint* (*the number of available islands*), *obtain a scheduled and bound DFG with minimized latency targeting DRFM-IID*.

The overall flow of the proposed method is shown in Fig. 6. Given a DFG, list-scheduling is first employed to obtain an initial scheduling result. Then *island assignment* and *iterative latency minimization* are applied consecutively to get a final solution. Island assignment is performed to

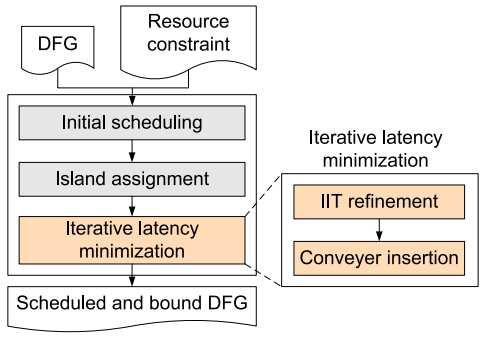


Fig. 6 The overall flow of the proposed algorithm.

bind operations into islands considering both the number of IITs and the timing-criticality of IIT. Then, iterative latency minimization tries to iteratively reduce the number of IITs, balance island utilization, and perform necessary conveyer insertion to preserve data dependency. There are two essential steps in iterative latency minimization: *IIT refinement* and *conveyer insertion*. At the k-th iteration, the former is responsible for not only IIT reduction but also island utilization balancing, while the latter is used to keep data dependency intact. At the end of the iteration, the partially scheduled and bound DFG (from the first to the k-th control step) are fixed, and the procedure then moves to the next iteration. More details are revealed in the remainder of this section.

4.1 Island Assignment

Island assignment is conducted to bind operations into islands. As previously mentioned, IIT count and timing-criticality of IIT should be the major concerns. Hence an edge-weighted compatibility graph $H(V, F)$ is built from the scheduled DFG $G(V, E)$ after list scheduling. The vertex set of H is the same as the one in G, whereas the edge set F is a superset of E. There are two types of edges in F: those that belong to E and those that do not. An edge belonging to F but not to E connects a pair of vertices that have no data dependency but are timing-compatible and thus can be assigned to the same island. Weights on those two types of edges are defined in (2), where the term $cri(e_{i,j})$ suggesting the timing-criticality of the corresponding transfer is given in (3).

$$W(e_{i,j}) = \begin{cases} 0, & e_{i,j} \notin E \\ 1 + cri(e_{i,j}), & e_{i,j} \in E \end{cases} \tag{2}$$

$$cri(e_{i,j}) = \frac{1}{cstep(v_j) - cstep(v_i)} \tag{3}$$

After properly setting edge weights, the island assignment problem can be formulated as finding a set of flows with maximum sum of weights in the compatibility graph, i.e., the min-cost flow problem. This problem can then be optimally solved by a polynomial-time algorithm described in [28]. The proposed method tries to prevent timing-critical transfers from crossing islands as well as minimizes the IIT

count at the same time.

4.2 Iterative Latency Minimization — IIT Refinement

IIT refinement considers not only the number of IITs but also the balance of island utilization. Before going into details, a *maximum utilization island set* (*MUIS*) is defined first. An *MUIS* is a set of islands with maximum utilization. For example, as in Fig. 5(a) and Fig. 5(c), *MUIS* is $\{I_A, I_C\}$ and $\{I_C\}$, respectively.

The IIT refinement process is based on the KL algorithm [29], which is broadly used in partitioning-related problems. During the process, operation nodes and bubbles are swapped for IIT minimization. A swap can be made between two nodes or between a node and a bubble. More precisely, a swap attempt can only be made between two feasible candidates over all islands. A node is considered feasible only on following conditions: i) it must be unlocked, ii) none of its immediate predecessors comes from a conveyer, and iii) it must be in the current cstep. Meanwhile, a bubble is considered feasible only if it is in the current cstep. The gain of a swap pair (u, v) is denoted as $g_{u,v}$ and given in (4), where T represents the number of IITs that can be reduced and D, defined in (5), indicates the difference of the average number of in-edges incident to islands in *MUIS* before and after the swap. Meanwhile, α in (4) is a user-configurable parameter and is set to 10 in all of our experiments shown later.

$$g_{u,v} = D + \alpha \cdot T \tag{4}$$

$$D = \frac{\sum_{I_i \in MUIS_{old}} \#in\text{-}edges_{old}(I_i)}{|MUIS_{old}|} - \frac{\sum_{I_i \in MUIS_{new}} \#in\text{-}edges_{new}(I_i)}{|MUIS_{new}|} \tag{5}$$

All feasible swap pairs over all islands are collected into a *feasible swap pair set* (*FSPS*). After performing a swap attempt, *FSPS*, *MUIS*, and gains of all swap pairs are updated accordingly. This swapping process is not terminated until *FSPS* is finally empty. Similar to the KL algorithm, only the first consecutive swaps with maximally positive accumulated gain sum are actually performed. The key steps of IIT refinement are summarized as follows:

i) Set all operation nodes unlocked.
ii) Find a swap pair with the largest gain from *FSPS*.
iii) Swap the pair then lock the operation node.
iv) Update *FSPS* and recalculate the gains of pairs in *FSPS*.
v) Repeat ii) to iv) until *FSPS* is empty.
vi) Keep the first k swaps and undo the rest if the accumulated gain sum of the first k swaps is the largest and positive; go to i).
vii) Otherwise, terminate IIT refinement.

All in all, the goal of IIT refinement is to minimize IIT count, to balance island utilization, and to reduce in-edges of highly-utilized islands at the same time.

4.3 Iterative Latency Minimization — Conveyer Insertion

After IIT refinement, conveyers may be inserted to preserve data dependency. Figure 7 shows the case when the procedure arrives at $cstep_4$. Note that an IIT does not need a conveyer in its destination island if the data item it carries is already available there. For example, as shown in Fig. 7, it is unnecessary to add a conveyer for $IIT_{3,2}$ since $IIT_{3,1}$ has already carried the same data item to I_A at $cstep_2$. Furthermore, an inserted conveyer does not always increase system latency because it tries to take over a bubble if there is any in the same island, as $IIT_{4,10}$ in Fig. 7. At the end of an iteration, all necessary conveyers are properly inserted before the currently working control step (i.e., $cstep_4$ in this case), and all data dependency remains intact. The procedure then moves to the next iteration.

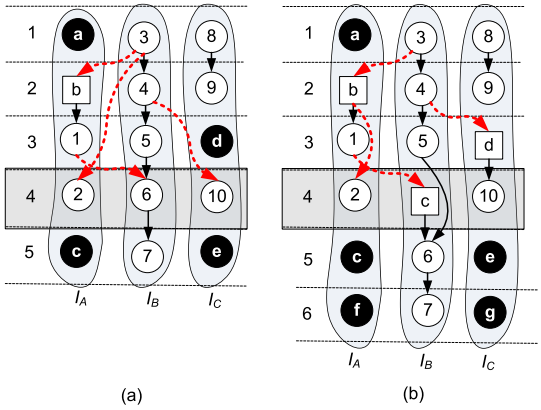


Fig. 7 The scheduled and bound DFG (a) before and (b) after conveyer insertion.

5. Experimental Results

5.1 Experimental Setup

The proposed algorithm has been implemented in C++/Linux environment and all experiments are conducted on a workstation with an Intel Xeon 3.2 GHz CPU and 4 GB RAM. For fair and comprehensive comparisons, two different synthesis flows are developed, as shown in Fig. 8. Given an input DFG and a resource constraint (i.e., the number of available islands), list scheduling is first performed to produce an initial scheduling result for both flows. Then, *Flow1* implements the approach proposed in [14], [15]. Since that approach does not consider inter-island transfer delay, conveyer insertion is thus mandatory as a post-processing step to ensure correct data dependency in DRFM-IID. *Flow2* precisely carries out the algorithm proposed in this article. The test cases are selected from several benchmark sets, which are frequently used in the high-level synthesis field [30]–[32]. The basic information of these test cases (DFGs) is reported in Table 1. The first three columns give names, numbers of nodes, and numbers of edges, respectively. The last column shows the latency (without inter-island delay consideration) obtained through ASAP scheduling, which indicates the lowest possible latency a synthesis framework can achieve.

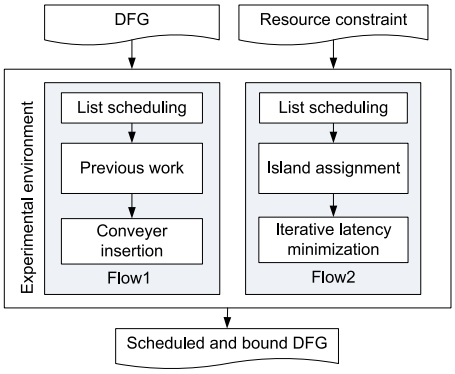


Fig. 8 The Experimental flows for comparisons.

Table 1 The basic information of the test cases.

Test case	#nodes	#edges	Latency
wang	48	58	7
lee	49	62	9
feedback	53	50	7
h2v2	53	54	17
fft4	62	88	8
cosine1	66	76	8
writebmp	106	88	7
matmul	109	116	9
smooth	197	196	11
invert	333	354	11

Two configurations are elaborated in our experiments — synthesis is performed without and with a resource constraint, respectively. In Configuration 1, the number of available islands is set as the minimum number that still guarantees the synthesis outcome with the minimum latency obtained from ASAP scheduling; i.e., there is in fact no resource constraint at all. Nevertheless, the assumption about unlimited available hardware resource is obviously impractical in the real world. Hence, in Configuration 2, for every test case the number of available islands is reduced by half; that is:

$$\#\text{ islands in Config. } 2 = \left\lceil \frac{\#\text{ islands in Config. } 1}{2} \right\rceil \quad (6)$$

5.2 Experimental Results and Discussions

The experimental results are given in Table 2 and Table 3 for Configuration 1 and Configuration 2, respectively. The latency values given in both tables apparently indicate that inter-island delay does greatly lengthen system latency. Table 2/3 reports that Flow1 requires 102.6%/97.2% and Flow2 requires 66.1%/43.3% more control steps on average, respectively. Therefore, it can be summarized that inter-island delay has a significant impact on system latency and must be handled with extreme care for every performance-driven architectural synthesis flow targeting DRFM-IID.

It is also concluded that the proposed framework

Table 2 The experimental results — unlimited #islands (Configuration 1).

Test case	#islands	Latency			Reduction	#IITs		Reduction	#IICs		Increment
		ASAP	Flow1	Flow2		Flow1	Flow2		Flow1	Flow2	
wang	8	7	16	12	25.00%	39	26	33.33%	17	21	23.53%
lee	6	9	17	14	17.65%	32	20	37.50%	11	14	27.27%
feedback	9	7	13	11	15.38%	27	18	33.33%	12	17	41.67%
h2v2	4	17	32	27	15.63%	24	17	29.17%	6	9	50.00%
fft4	8	8	21	16	23.81%	63	45	28.57%	24	34	41.67%
cosine1	9	8	16	15	6.25%	48	33	31.25%	20	29	45.00%
writebmp	16	7	13	11	15.38%	29	25	13.79%	18	26	44.44%
matmul	16	9	16	14	12.50%	39	38	2.56%	27	38	40.74%
smooth	27	11	21	16	23.81%	91	67	26.37%	49	63	28.57%
invert	36	11	24	19	20.83%	175	93	46.86%	77	96	24.68%
Avg.					17.62%			28.27%			36.76%

Table 3 The experimental results — constrained #islands (Configuration 2).

Test case	#islands	Latency			Reduction	#IITs		Reduction	#IICs		Increment
		ASAP	Flow1	Flow2		Flow1	Flow2		Flow1	Flow2	
wang	4	13	26	20	23.08%	35	24	31.43%	16	12	-25.00%
lee	3	18	36	25	30.56%	36	23	36.11%	6	6	0.00%
feedback	4	14	28	20	28.57%	31	14	54.84%	8	10	25.00%
h2v2	2	29	43	36	16.28%	16	13	18.75%	2	2	0.00%
fft4	4	16	37	29	21.62%	58	44	24.14%	10	12	20.00%
cosine1	4	17	34	24	29.41%	46	26	43.48%	8	11	37.50%
writebmp	8	14	24	19	20.83%	42	27	35.71%	14	24	71.43%
matmul	8	15	29	21	27.59%	68	37	45.59%	19	28	47.37%
smooth	13	17	37	23	37.84%	111	69	37.84%	32	65	103.13%
invert	18	20	42	28	33.33%	204	107	47.55%	63	94	49.21%
Avg.					26.91%			37.54%			32.86%

achieves an average reduction of 26.91% (17.62%) and 37.54% (28.27%) in latency and IIT count respectively as compared to the prior art [14], [15] with (without) a resource constraint. The experimental results clearly indicate that there is a strong link between the number of IITs and resultant system latency. Furthermore, besides shorter latency, fewer IITs in use can also imply less power consumption for on-chip communication [4], which is regarded as a desirable side effect.

Since the proposed method focuses on latency minimization, reduction of physical inter-island connects (IICs) is indeed not its primary goal. As a result, Flow1 does perform better than Flow2 in terms of IIC count because that is what Flow1 is exactly designed for. As a matter of fact, once we have attempted to incorporate the number of IICs into part of gain calculation as a secondary goal in the proposed iterative optimization process. Nevertheless, the actual IIC reduction is limited and thus not emphasized in this article. We believe what really happens here should be regarded as a typical area-delay (IIC-latency) tradeoff that can be found in virtually all synthesis problems.

6. Conclusion

In this article, we first propose a new distributed register-file based microarchitecture named DRFM-IID, which takes inter-island transfer delay into account. The corresponding synthesis task is inherently more complicated than those without inter-island delay consideration. Therefore, we develop a new performance-driven architectural synthesis framework. There are two major procedures in the proposed algorithm. First, island assignment performs operation binding while considering both IIT count and timing-criticality of data transfer. The problem is formulated as a min-cost network flow one and optimally solved accordingly. Then, iterative latency minimization composed of IIT refinement and conveyer insertion is conducted to validate and further improve a solution. IIT refinement is to minimize IIT count, to balance island utilization, and to reduce in-edges of highly-utilized islands, while conveyer insertion tries to keep data dependency intact without further increasing latency. The experimental results show that a reduction of 26.91% and 37.54% is achieved in latency and IIT count as compared to the prior art, where the latter is also an indicator for on-chip communication power consumption. Therefore, we believe the proposed method is a better synthesis solution for high-performance and low-power applications targeting DRFM-IID.

Acknowledgment

This work was supported in part by the National Science Council of Taiwan under Grant NSC 99-2220-E-009-008.

References

[1] International Technology Roadmap for Semiconductors, Semiconductor Industry Association, 2009.

[2] D. Matzke, "Will physical scalability sabotage performance gains?," Computer, vol.30, no.9, pp.37–39, 1997.

[3] L.P. Carloni and A.L. Sangiovanni-Vincentelli, "Coping with latency in SOC design," IEEE Micro, vol.22, no.5, pp.24–35, 2002.

[4] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, "Interconnect-power dissipation in a microprocessor," Proc. Int'l Workshop System Level Interconnect Prediction, pp.7–13, 2004.

[5] A. Singh, G. Parthasarathy, and M. Marek-Sadowska, "Efficient circuit clustering for area and power reduction in FPGAs," ACM Trans. Design Automation of Electronics Systems, vol.7, no.4, pp.643–663, Oct. 2002.

[6] E. Kusse and J. Rabaey, "Low-energy embedded FPGA structures," Proc. Int'l Symp. Low Power Electronics and Design, pp.155–160, 1998.

[7] T. Tuan and B. Lai, "Leakage power analysis of a 90 nm FPGA," Proc. IEEE Custom Integrated Circuits Conf., pp.57–60, 2003.

[8] D.M. Chapiro, Globally-asynchronous locally-synchronous systems, Ph.D. dissertation, Stanford Univ., Stanford, CA, 1984.

[9] E. Beigne and P. Vivet, "Design of on-chip and off-chip interfaces for a GALS NoC architecture," Proc. Int'l Symp. on Asynchronous Circuits and Systems, pp.174–183, 2006.

[10] L.P. Carloni, K.L. McMillan, A. Saldanha, and A.L. Sangiovanni-Vincentelli, "A methodology for correct-by-construction latency insensitive design," Proc. Int'l Conf. Computer Aided Design, pp.309–315, 1999.

[11] D. Bufistov, J. Júlvez, and J. Cortadella, "Performance optimization of elastic systems using buffer resizing and buffer insertion," Proc. Int'l Conf. Computer Aided Design, pp.442–448, 2008.

[12] J.-D. Huang, Y.-S. Huang, L. Wang, and G.-W. Lee, "Throughput-aware floorplanning via dynamic optimization on performance-critical loops," Int'l Journal of Electrical Engineering, vol.17, no.1, pp.33–42, 2010.

[13] J.-D. Huang, Y.-H. Chen, and Y.-C. Ho, "Throughput optimization for latency-insensitive system with minimal queue insertion," Proc. Asia and South Pacific Design Automation Conf., pp.585–590, Jan. 2011.

[14] J. Cong, Y. Fan, and W. Jiang, "Platform-based resource binding using a distributed register-file," Proc. Int'l Conf. Computer Aided Design, pp.709–715, 2006.

[15] J. Cong, Y. Fan, and J. Xu, "Simultaneous resource binding and interconnection optimization based on a distributed register-file microarchitecture," ACM Trans. Design Automation of Electronics Systems, vol.14, no.3, pp.1–31, May 2009.

[16] K. Lim, Y. Kim, and T. Kim, "Interconnect and communication synthesis for distributed register-file microarchitecture," Computers & Digital Techniques, IET, vol.3, no.2, pp.162–174, March 2009.

[17] J.-D. Huang, C.-I Chen, Y.-T. Lin, and W.-L. Hsu, "Communication synthesis for interconnect minimization targeting distributed register-file microarchitecture," IEICE Trans. Fundamentals, vol.E94-A, no.4, pp.1151–1155, April 2011.

[18] J.-D. Huang, C.-I Chen, W.-L. Hsu, Y.-T. Lin, and J.-Y. Jou, "Performance-driven architectural synthesis for distributed register-file microarchitecture considering inter-island delay," Proc. Int'l Symp. VLSI Design, Automation and Test, pp.169–172, April 2010.

[19] D. Kim, J. Jung, S. Lee, J. Jeon, and K. Choi, "Behavior-to-placed RTL synthesis with performance-driven placement," Proc. Int'l Conf. Computer Aided Design, pp.320–325, 2001.

[20] J. Cong, Y. Fan, G. Han, X. Yang, and Z. Zhang, "Architecture and synthesis for on-chip multicycle communication," IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst., vol.23, no.4, pp.550–564, April 2004.

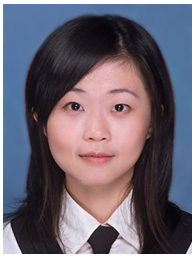
[21] C.-I Chen and J.-D. Huang, "A hierarchical criticality-aware architectural synthesis framework for multicycle communication," IEICE Trans. Fundamentals, vol.E93-A, no.7, pp.1300–1308, July 2010.

[22] S.-H. Huang, C.-H. Chiang, and C.-H. Cheng, "Three-dimension scheduling under multi-cycle interconnect communications," IEICE Electronics Express, vol.2, no.4 pp.108–114, Feb. 2005.

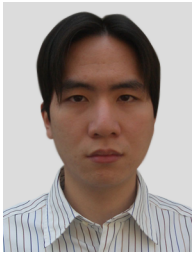
[23] A. Ohchi, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "High-level synthesis algorithms with floorplaning for distributed/shared-register architectures," Proc. Int'l Symp. VLSI Design, Automation and Test, pp.164–167, April 2008.
[24] S. Gao, K. Seto, S. Komatsu, and M. Fujita, "Pipeline scheduling for array based reconfigurable architectures considering interconnect delays," Proc. Int'l Conf. Field-Programmable Technology, pp.137–144, Dec. 2005.
[25] J. Cong, Y. Fan, and Z. Zhang, "Architecture-level synthesis for automatic interconnect pipelining," Proc. Design Automation Conf., pp.602–607, June 2004.
[26] W.-S. Huang, Y.-R. Hong, J.-D. Huang, and Y.-S. Huang, "A multicycle communication architecture and synthesis flow for global interconnect resource sharing," Proc. Asia and South Pacific Design Automation Conf., pp.16–21, Jan. 2008.
[27] Y.-S. Huang, Y.-J. Hong, and J.-D. Huang, "Communication synthesis for interconnect minimization in multicycle communication architecture," IEICE Trans. Fundamentals, vol.E92-A, no.12, pp.3143–3150, Dec. 2009.
[28] R. Ahuja, T. Magnanti, and J. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, 1993.
[29] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," Bell Syst. Tech. J., pp.291–307, Feb. 1970.
[30] MCAS: multicycle architectural synthesis system. [Online]. Available: http://cadlab.cs.ucla.edu/software_release/mcas/
[31] ExPRESS group. [Online]. Available: http://express.ece.ucsb.edu/
[32] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms, 2nd ed., the MIT Press, 2001.

Juinn-Dar Huang received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1992 and 1998, respectively. He is currently an Associate Professor in the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University. His research interests include behavioral and logic synthesis, design verification, 3D IC architecture/CAD, and microprocessor design. He recently served in the Organizing Committees of IEEE/ACM ASP-DAC 2010 and SASIMI 2010. He was the Secretary General of Taiwan IC Design Society (TICD) from 2004 to 2008, and the Technical Program Committee Vice-Chair of VLSI Design/CAD Symposium 2008. He served as a Technical Program Committee member of IEEE/ACM DATE (2008, 2010), and IEEE VLSI-DAT (2010, 2011). He was also the Organizing Committee member of IEEE ICFPT 2008. He is a member of the IEEE, ACM, and Phi Tau Phi.

Chia-I Chen received the B.S. degree in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2005, where she is currently working toward the Ph.D. degree in the Institute of Electronics. Her current research interests include high-level synthesis and computer architecture.



Wan-Ling Hsu received the B.S. degree in electrical engineering from Chang Gung University, Taoyuan, Taiwan, in 2007, and the M.S. degree in electronics engineering, National Chiao Tung University, Hsinchu, Taiwan, in 2009. Her research interests include high-level synthesis and physical synthesis.



Yen-Ting Lin received the B.S. and the M.S. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2007 and 2009, respectively. His research interests include high-level synthesis and digital VLSI design.

Jing-Yang Jou received the B.S. degree in electrical engineering from National Taiwan University, Taiwan, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign, in 1979, 1983, and 1985, respectively. He is the Deputy Minister of National Science Council, Taiwan from February 2010. He was the Vice Chancellor (Academic Affairs), University System of Taiwan (consisting of four research universities: National Central University, National Chiao Tung University, National Tsing Hua University, and National Yang Ming University) from April 2007 to January 2010, and the Executive Director of National SoC Program from April 2007 to January 2010. He is a Distinguished Full Professor and was Chairman of Electronics Engineering Department from 2000 to 2003 at National Chiao Tung University, Hsinchu, Taiwan. His research interests include logic and physical synthesis, design verification, CAD for low power and Network on Chips. He has published more than 160 technical papers. Dr. Jou is a Fellow of IEEE. He was elected to the President of the Taiwan Integrated Circuit Design Society (TICD) 2007–2008, and he serves as Associate Editor for IEEE Transactions on Very Large Scale Integration Systems from 2007 to present.