A Local-Conscious Global Register Allocator for VLIW DSP Processors with Distributed Register Files

Chia-Han Lu, Yung-Chia Lin, Yi-Ping You, and Jenq-Kuen Lee

Department of Computer Science,
National Tsing-Hua University,
Hsinchu 30013, Taiwan.
`chlu, yclin, ypyou, jklee@pllab.cs.nthu.edu.tw`

Abstract. Embedded processors developed in recent years have attempted to employ novel hardware design to reduce ever-growing complexity, power dissipation, and die area. While using a distributed register file architecture with irregular accessing constraints is considered to be an effective approach rather than traditional unified register file structures, conventional compilation techniques are not adequate to utilize such new register file organizations for optimal performance. This paper presents a novel scheme for register allocation which composes of global and local register allocation, on a VLIW DSP processor with distributed register files whose port access is highly restricted. In the scheme, a sub-phase prior to original global/local register allocation, named global/local RFA (register file assignment), is introduced to minimize various register file communication costs. For featured register file structure where each cluster contains heterogeneous register files, conventional register allocation scheme with cluster assignment only have to be enhanced to cope both inter-cluster and intra-cluster communications. Due to potential but heavy influences of global RFA on local RFA, a heuristic algorithm is proposed where global RFA manages to make suitable decisions on communication for local RFA. Experiments were done with a developing compiler based on the Open Research Compiler (ORC), and the results indicate that the compilation with the proposed approach delivers significant performance improvement, comparable to the solution using only the PALF scheme developed in our previous work.

1 Introduction

Embedded processors developed in recent years have attempted to employ novel hardware design to reduce ever-growing complexity, power dissipation, and die area. While using a distributed register file architecture with irregular accessing constraints is considered to be an effective approach rather than traditional unified register file structures, conventional compilation techniques are not adequate to utilize such new register file organizations for optimal performance. In our research work, we work on addressing the compiler schemes for a VLIW DSP processor with distributed register files, known as PAC architectures.

The appearances of multi-banks of register files, distributed register clusters, and ping-pong architectures on embedded VLIW DSP processors such as PAC architectures present a great challenge for compilers to generate efficient codes for multimedia applications. In the literature, current research results in compiler optimizations for such problems have been limited to address issues for cluster-based architectures. It includes the work on partitioning register files to work with instruction scheduling [18], loop partitions for clustered register files [19], and cluster register files [16]. The work in [15] begins to address this complex optimization issue for embedded DSP processors, but only in the layer of copy propagation optimizations, and the work in [20] attempts to deal with software pipelining issues with distributed register files. In this paper, we address the issues dealing with global register allocations.

This work is for the compiler work based on Parallel Architecture Core (PAC) DSP [3,7,8], which is designed with distinctively banked register files where port access is highly restricted. The PAC DSP employs a heterogeneous design that equips one singular scalar unit (for light-weight arithmetic, address calculation, and program flow control), plus two data stream processing clusters in which each one contains a pair of load/store unit and ALU/MAC unit with powerful SIMD capabilities; every unit in the clusters collocates three varied types of register files, providing different accessing manners and constraints, and the scalar unit has its own accessible register file deployed. The major specialty of the register file architectures featured by the PAC DSP processor is that it incorporates a so-called *ping-pong register file structure* [6], which is divided into two banks and in which banks can only be restrictedly accessible in a mutual-exclusive way, as a semi-centralized register file among clusters and functional units within a cluster. With this design to decrease the power consumption because of fewer port connections, not only does the clustered design make register access across clusters an additional issue, but the switched access nature of the ping-pong register file raises our interest in investigating further register assignment to increase instruction level parallelism.

This paper presents a novel scheme for register allocation which composes of global and local register allocation, on a VLIW DSP processor with distributed register files whose port access is highly restricted. In the scheme, a sub-phase prior to original global/local register allocation, named global/local RFA (register file assignment), is introduced to minimize various register file communication costs. Our work is based on ORC software framework and in the framework, register allocation composes of global register allocation (GRA) and local register allocation (LRA). There are two kinds of temporary name (TN). One is local TN, while the other one is global TN. LRA manages to allocate register for local TN, whose liveness is bounded within a single basic block, GRA does for GTN, whose liveness will cross basic blocks. GRA proceeds before LRA in the back-end of ORC. For featured register file structure where each cluster contains heterogeneous register files, conventional register allocation scheme with cluster assignment only have to be enhanced to cope both inter-cluster and intra-cluster communications. Due to potential but heavy influences of global RFA on local

RFA, a heuristic algorithm is proposed where global RFA manages to make suitable decisions on communication for local RFA. Experiments were done with a developing compiler based on the Open Research Compiler (ORC), and the results indicate that the compilation with the proposed approach delivers significant performance improvement, comparable to the solution using only the PALF scheme developed in our previous work.

The remainders of this paper are organized as follows. In Section 2, we will introduce processor architecture and register file structure of PAC VLIW DSP. Proposed register allocation scheme involving register file assignment (RFA) will be covered in Section 3. Next, Section 4 provide comparison among register allocation scheme. Then, Section 5 presents related work on cluster assignment and our previous work on the processor. At last, Section 6 concludes the paper.

2 PAC DSP Architecture

The PAC DSP originally features a clustered VLIW architecture which boosts scalability, and a large number of registers which are arranged as innovative heterogeneous and distinct partitioned register file structures. Being unlike symmetric architectures of most DSP processors available nowadays, the PAC DSP processor is constructed as a heterogeneous five-way issue VLIW architecture, comprised of two integer ALUs (I-unit), two memory load/store units (M-unit), and the program sequence control unit/scalar unit (B-unit) which is mainly in charge of control flow instructions like branch and jump. Each unit has its own executable subset of instruction set and each executable instruction has its own register accessibility and constraints. The M- and I-units are organized in pairs, and each pair contains exactly one M-unit and one I-unit to form a cluster arrangement with associated register files. It is apparent that each cluster is logically appropriate for one data stream processing, and the current design of PAC DSP consists of two clusters to support maximum workload capacity of two concurrent data stream. But the scalability of the cluster design in PAC DSP could allow the processor to easily involve more clusters to handle larger data processing workload demand. The B-unit consists of two sub-components, the program sequence control unit, and the scalar unit, due to the hierarchical decoder design for variable-length instruction encoding in PAC DSP. The program sequence control unit primarily takes charge of operations of control flow instructions. The scalar unit, which is capable of simple load/store and address arithmetic, is placed separately from data stream processing clusters, with its own register file. The overall architecture is presented in Fig. 1.

Registers in PAC DSP are organized into several distinct partitioned register files and placed as cluster structures, to reduce wire connections between functional units and registers so that chip area and power consumption may be decreased. All units in the processor have their dedicated local register files attached: they include R-register file, AC-register file, and A-register file, which are only accessible by B-unit, I-unit, and M-unit, respectively. In each cluster comprised of M-unit and I-unit, a global register file named as D-register file

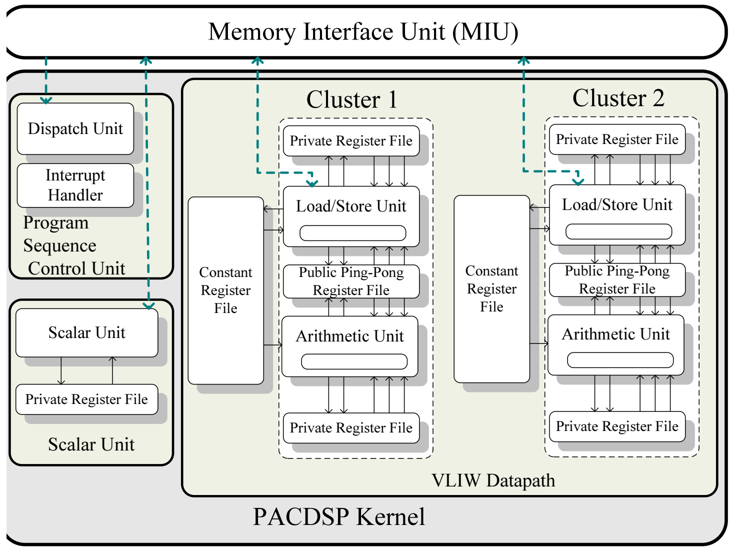

Fig. 1. PAC DSP architecture

is designed to be shared by the pair of M- and I-units in each cluster. The internal of the D register file is further partitioned into two banks to utilize the instructional port switching technology in order to reduce more wire connections between the M- and I-units. This technology, being referred to the name as 'ping-pong register file structure', is that decreasing the register bank port connection which limits the accessibility of the two bank; in each cycle, the two functional units can only access to different banks. Each instruction bundle encodes the information of which bank is to be accessed for each functional unit in the cycle so that the hardware can do port switching between D-register file banks and functional units, to attain the purpose of data sharing within a cluster. By using the concept of overlapping two different data-stream operations in a cluster, we may minimize the occasion that M- and I-units access the same data at the same time; therefore, the access constraints of 'ping-pong register file structure' should cause little impact on performance. The advantage of such a 'ping-pong register file structure' design is believed to consume less power due to its reduced read/write ports [11] while retaining the data communication capability. Besides local register files and global register files, each cluster contains an additional constant register file which is shared by both M- and I-units as one of the read-only operand sources usable by certain instructions. Only M-units can initialize the data in the constant register file.

3 Local-Conscious Global Register Allocation Scheme

This section describes our proposed local-conscious global register allocation scheme, including the adaptation of our previously proposed PALF local register allocation scheme.

3.1 Register File Assignment Phases

The most variance between the register allocation for conventional unified register file architectures and the the irregular distributed register file structures used in the PAC DSP is the variety of the register files which may be allocatable for each variable operated by an instruction. With the irregular designs and various port access constraints in PAC DSP, the proper register file to be allocated can not be easily determined to produce the optimized results since the allocation heavily interferes with the instruction scheduling. Therefore, we prefer to separate the register file assignment (RFA) from the register allocation, to simply the analysis and design of the proposed schemes, as well as the implementation to fit appropriately for the phase-ordering of our compiler infrastructures. The overall register allocation scheme currently involved in our developing compilers for PAC DSP is shown in Fig. 2,

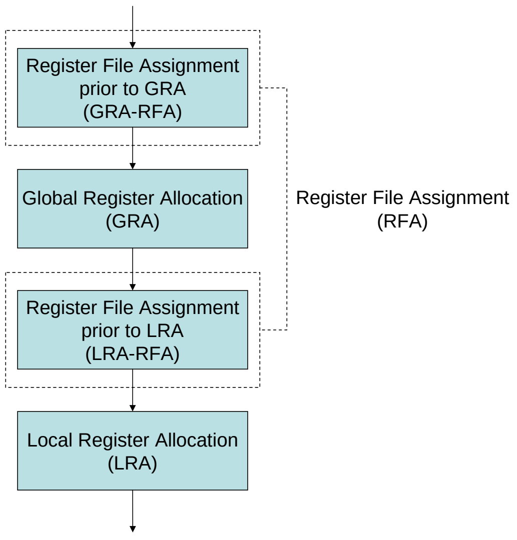
Register File Assignment prior to GRA (GRA-RFA)
Global Register Allocation (GRA)
Register File Assignment prior to LRA (LRA-RFA)
Local Register Allocation (LRA)
Register File Assignment (RFA)

Fig. 2. Register allocation scheme involving register file assignment (RFA)

3.2 Ping-pong Aware Local Favorable Register Allocation (PALF)

This section reviews our previously proposed register allocation algorithm which, given a dependency DAG (Directed Acyclic Graph) [1] that describes the compilation regions, heuristically determines the proper register file/bank assignment and employ state-of-the-art graph-coloring register allocation for each assigned register file/bank in PAC architectures.

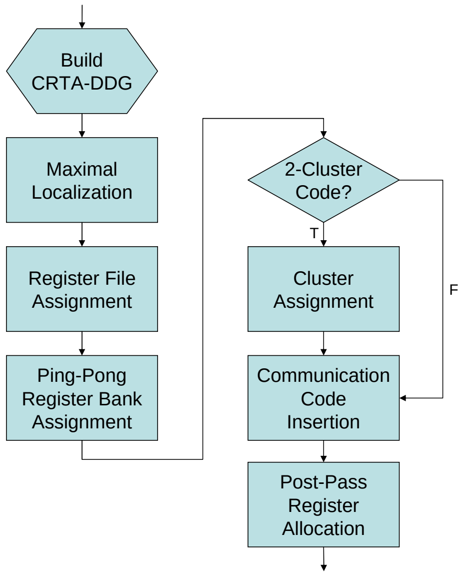

Fig. 3. Flowchart of PALF

The overall register allocation algorithm proposed is shown in Fig. 3. Our approach requires building an extended data dependence DAG, which preserves the information of the execution and storage relationship for irregular constraint analysis, in addition to the original partial order imposed by instruction precedence constraints. Nodes in the extended data dependence DAG represent instructions of the input code block, with the component-type association (that indicates which functional unit is preferred to be scheduled for this node) and the register-type association (that annotates the appreciated physical register file/bank, to where the operands/results will be allocated); the edges linked between the nodes represent data dependency that serializes the execution order to be followed in the scheduled code sequence. The main PALF register allocation scheme could be organized into five phases as follows:

1. Perform the preferable functional unit assignment to the default execution type of each instruction by the "maximal localization" analysis.

2. Assign operands/results (required to be allocated to physical registers) of each node in the extended data dependence DAG to the desirable register files.
3. Partition the operands/results assigned to the global ping-pong register files to the preferred register banks by the strategy of optimizing ping-pong parallelism.
4. Optionally partition the nodes in the extended data dependence DAG into two clusters properly if we would like to compile for two-clusters.
5. Insert nodes of required communication code to avoid invalidity caused by the register file/bank assignment and cluster-partitioning, followed by the physical register allocation for each register file.

3.3 Extending RFA for Global Register Allocation

Due to the variety of distributed register files involved and irregular access constraints in the design of PAC DSP architectures, we developed the PALF scheme to include an appropriate register file assignment phase in our developing compilers for PAC DSP before the actual local register allocation, allowing us to employ state-of-the-art graph-coloring based methods for allocating homogeneous registers in the same register file. This design simplifies the implementation of optimizing the register allocation for the irregular and distributed register files used in PAC DSP but still achieves the expected performance.

While the proposed PALF scheme performs excellently with the local optimizations and instruction scheduling, the global register allocation cannot be directly derived from the PALF since the original design decisions in the heuristics is based on the data dependence DAG. Therefore, we add a new phase of register file assignment for global register allocation (GRA-RFA), which refers to the local preference from the PALF scheme for each basic block and determines the proper RFA for GTNs across several blocks. Our proposed GRA-RFA extends the PALF scheme to conform the RFA of GTNs to the results that are advantageous for both LRA-RFA and local instruction scheduling. On top of the performance-sensitive properties of LRA-RFA, we establish the GRA-RFA by the following procedures:

1. Employ a pseudo PALF-RFA to get the full localized assignment for each GTN and TN if localizing all GTNs in each basic blocks, and let these RFA be the base reference.
2. Build the GTN-TN interference map for each GTN, and follow the access constraints of instructions and register files to obtain an initial RFA properly for each GTN, regardless of the correlation between each GTN that may be across several basic blocks.
3. Prioritize basic blocks by several factors. The factors may include the execution frequency from the profiling, loop depth, pseudo schedule length, numbers of operations, and so on. This process will rely on the appropriate cost models constructed by these factors.

4. Progressively optimize the GTN-RFA by the sequence of the priority obtained in the third procedure. From the most prioritized basic block to the least one, we apply the pseudo PALF scheme again to see any adjustment of GTN-RFA is needed if two of GTNs in the same basic block may produce the improper penalty of register file communication or inefficient results.

Although the full development of the proposed GRA scheme is still in progress, we currently use a reduced version of the GRA-RFA, which bypasses the first two procedures and employs the one-pass pseudo PALF-RFA for each basic block using the greedy policy of the priority, to determine the assignment of a GTN by its involved basic block with the highest priority.

3.4 An Example of Local-Conscious GRA-RFA

In this section we show a simple example to illustrate the current version of the proposed GRA-RFA scheme. Fig. 4 presents the control flow of our example, where basic block is omitted as BB. It starts with BB11 and ends with BB12. There are two loops. One contains only BB2, while the other comprises BB4, BB5, BB6, and BB7.

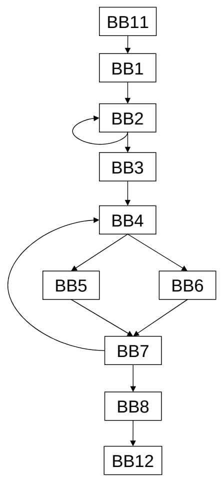

Fig. 4. The control flow graph of basic blocks in the example

We use Fig. 5 to exhibit the scheme flows and the possible results. Each number in the most left column presents a GTN. In the example, index of GTNs is bounded between 134 and 170 and there are 17 GTNs in total. On the other side, the most top row includes all basic blocks involved in the example, except entry/exit basic blocks, BB11 and BB12. Then the number in parentheses presents the priority of basic block determined in the proposed GRA scheme, while the smaller number means the higher priority. The rest elements which represent register files compose the whole process of the scheme and the last column labeled with "Assigned" summaries the final register file assignment.

	BB1 (8)	BB2 (3)	BB3 (6)	BB4 (4)	BB5 (1)	BB6 (2)	BB7 (5)	BB8 (7)	**Assigned**
GTN134	A1	**A1**							**A1**
GTN137	A1	**A1**							**A1**
GTN138		**A1**							**A1**
GTN139	AC2	**AC2**							**AC2**
GTN148			D1		**D1**			D1	**D1**
GTN150			D1		**D1**			D1	**D1**
GTN152			D1		**D1**			D1	**D1**
GTN154			D1		**D1**			D1	**D1**
GTN156			D1			**D1**		D1	**D1**
GTN158			D1			**D1**		D1	**D1**
GTN160			D1			**D1**		D1	**D1**
GTN162			D1		**D1**	D1			**D1**
GTN164			D1		**D1**	D1			**D1**
GTN166			D1			**D1**		D1	**D1**
GTN168			AC1	**AC1**			AC1		**AC1**
GTN169			D1			**D1**		D1	**D1**
GTN170			AC1		**AC1**			D1	**AC1**

Fig. 5. An example of global extension of PALF

In Fig. 5, the process of the scheme begins from BB5 with the highest priority. The pseudo PALF-RFA is applied on the basic block and GTN160 in BB5 is assigned to AC1 and the rest in BB5 are assigned to D1. According to our algorithm, the register file assignment of these GTNs is designated and will not allow to be altered in the later processes. Then, since BB5 has been done, the highest priority goes to BB6. In BB6, for there are two GTNs whose register files are fixed previously, the pseudo PALF-RFA uses it as its precondition when it is applied to BB6. After the process of BB6, register files of GTN156, GTN158, GTN160, GTN166, and GTN169 are assigned to D1. Finally, after the next two passes, those of GTN134, GTN137, GTN138, GTN139, and GTN168 are assigned. Because register files of all GTNs have been assigned, the remain passes will not alter the register file assignment of each GTN anymore.

4 Experiment and Discussion

This section describes our preliminary experiments on the proposed GRA scheme. Fig. 6 shows the comparison of various register allocation schemes performed on the DSPstone benchmark suite [13] in the PAC DSP, which are implemented on our developing compilers based on ORC infrastructures and evaluated with a cycle-accurate instruction set simulator of PAC DSP.

Here are the three register allocation schemes to be compared, the naïve code generation without any optimization on register allocation, PALF register allocation scheme, and PALF register allocation scheme plus the proposed GRA scheme. The comparison indicates that the PALF one plus the proposed GRA scheme can provide 16.9% speedup on average when comparing to the naïve one. In these preliminary experiments basic blocks under the proposed scheme are prioritized simply based on the length and depth of loop, while other potential factors may include the schedule length, density of GTN, etc. It would require more further investigations to build a better prioritizing function for the scheme.

Most of the programs in DSPstone benchmark suite have fine speedup using the scheme, except *complex_multiply*. *complex_multiply* reveals another issue in developing compilers for the PAC DSP architecture which features irregular distributed register file structure. For *complex_multiply*, some register files are highly utilized in the proposed GRA and lead to a lot of spilling in the LRA. And such an anomaly degrades performance in the benchmark. To solve this, register pressure control in every register file will be required in RFA to provide proper apportionment between LRA and GRA. Since the number of registers in register files on PAC DSP is limited, the control relies on the precise modelings of potential register usage scenarios.

5 Related Work

Hiser, Carr, and Sweany present global register partitioning for cluster assignment in [16], where register component graph (RCG) built on data dependence DAGs (DDD)s and optimal schedule is used to model relationship between registers. Every node in RCG represents register operands and edge weight represents 'affinity' between nodes the edge connects. Positive weight indicates related nodes should be put in the same cluster, while negative one otherwise. They use greedy heuristic algorithm to divide these nodes into two clusters. Experimental result shows their algorithm has only about 10% performance degradation. However, their architecture is simpler than PAC DSP which has ping-pong constraint. Problem in PAC DSP can't be reduced simply into the algorithm.

Terechko, Thénaff, and Corporaal provide another for cluster assignment of global values in [17]. Local values have short live ranges, while global values can be alive throughout a whole programming unit. In the paper, they provided several kinds of ideas how to do the task. One of them is to use 'affinity' mentioned above. Affinity between two global values indicates the benefit of assigning them to the same cluster based on the data flow graph. Instead of RCG mentioned above in [16], they use equations to present affinity between two global values. At last, they have detailed experimental result on their ideas to give comparison among them.

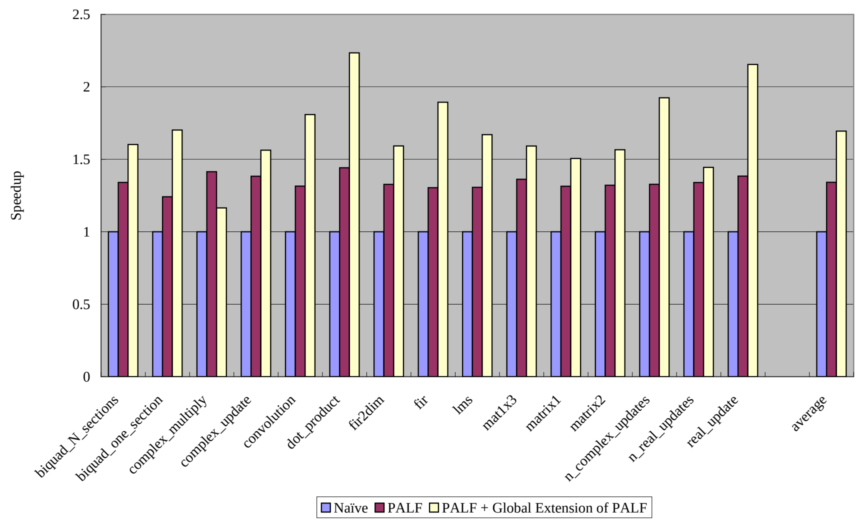

Fig. 6. The comparison between the register allocation schemes

Besides PALF, our prior work also proposed a simultaneous instruction scheduling/register file assignment method based on simulated annealing (SA) [9]. The SA method iteratively optimizes the entire RFA state of a single basic block, and uses the instruction scheduler itself as the evaluation function; an approach borrowed from [18]. This local method can also be globally extended in the same manner as PALF as presented in this paper. This may be an item for future work.

In addition to work on register allocation scheme, modification of copy propagation in original ORC is proposed in [15]. Due to irregular distributed register file structure in PAC DSP, conventional copy propagation might degrade performance. In the paper, a communication cost model is derived for copy propagation, which is build on cluster distance, register port pressure, and movement type of register sets. The model is used to guide data flow analysis for better performance on PAC DSP architecture. This scheme is effective to prevent performance anomaly.

6 Conclusion

This paper proposed a new scheme involving separated phases named register file assignment from the typical register allocation, which includes two sub-phases, LRA-RFA and GRA-RFA. Due to the irregular distributed register file structures on PAC DSP architectures, where the conventional register allocation is not appropriate, register file assignment could become the core procedure of the scheme to boost performance on the architecture. Preliminary experimental results showed that the proposed scheme can utilize the distributed register file architectures and deliver great performance. Our future work will include the complete exploration of the various issues involved in our proposed scheme and the full version of the implementations.

References

1. Aho, A. V., J. D. Ullman, and R. Sethi: Compilers Principles, Techniques, and Tools. Addison-Wesley, Reading, MA, 1986.
2. CEVA: CEVA-X1620 Datasheet. CEVA, 2004.
3. David Chang and Max Baron: Taiwan's Roadmap to Leadership in Design. Microprocessor Report, In-Stat/MDR, Dec. 2004. http://www.mdronline.com/mpr/archive/mpr_2004.html
4. A. Capitanio, N. Dutt, and A. Nicolau: Partitioned Register Files for VLIW's: A Preliminary Analysis of Tradeoffs. Proceedings of the 25th Annual International Symposium on Microarchitecture (MICRO-25), pages 292–300, Portland, OR, December 1–4 1992.
5. R. Ju, S. Chan, and C. Wu: Open Research Compiler for the Itanium Family. Tutorial at the 34th Annual Int'l Symposium on Microarchitecture, Dec. 2001.
6. T.-J. Lin, C.-C. Lee, C.-W. Liu, and C.-W. Jen: A Novel Register Organization for VLIW Digital Signal Processors. Proceedings of 2005 IEEE International Symposium on VLSI Design, Automation, and Test, pages 335–338, 2005.
7. T.-J. Lin, C.-C. Chang. C.-C. Lee, and C.-W. Jen: An Efficient VLIW DSP Architecture for Baseband Processing. Proceedings of the 21th International Conference on Computer Design, 2003.
8. Tay-Jyi Lin, Chie-Min Chao, Chia-Hsien Liu, Pi-Chen Hsiao, Shin-Kai Chen, Li-Chun Lin, Chih-Wei Liu, Chein-Wei Jen: Computer architecture: A unified processor architecture for RISC & VLIW DSP. Proceedings of the 15th ACM Great Lakes symposium on VLSI, April 2005.
9. Yung-Chia Lin, Chung-Lin Tang, Chung-Ju Wu, Ming-Yu Hung, Yi-Ping You, Ya-Chiao Moo, Sheng-Yuan Chen, and Jenq Kuen Lee: Compiler Supports and Optimizations for PAC VLIW DSP Processors. Proceedings of the 18th International Workshop on Languages and Compilers for Parallel Computing, 2005.
10. R. A. Ravindran, R. M. Senger, E. D. Marsman, G. S. Dasika, M. R. Guthaus, S. A. Mahlke, and R. B. Brown: Increasing the number of effective registers in a low-power processor using a windowed register file. Proceedings of the 2003 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '03), 125–136, 2003.
11. S. Rixner, W. J. Dally, B. Khailany, P. Mattson, U. J. Kapasi, and J. D. Owens: Register organization for media processing. International Symposium on High Performance Computer Architecture (HPCA), pp.375-386, 2000.
12. Texas Instruments: TMS320C64x Technical Overview. Texas Instruments, Feb 2000.

13. V. Zivojnovic, J. Martinez, C. Schlager, and H. Meyr: DSPstone: A DSP-oriented benchmarking methodology. Proceedings of the International Conference on Signal Processing and Technology, 715–720, 1995.
14. Yung-Chia Lin, Yi-Ping You, and Jenq-Kuen Lee: Register Allocation for VLIW DSP Processors with Irregular Register Files. Proceedings of the 12th Workshop on Compilers for Parallel Computers (CPC 2006), Jan 9–11 2006.
15. Chung-Ju Wu, Sheng-Yuan Chen, and Jenq-Kuen Lee: Copy Propagation Optimizations for VLIW DSP Processors with Distributed Register Files Proceedings of the 19th International Workshop on Languages and Compilers for Parallel Computing, Nov. 2–4 2006.
16. Jason Hiser, Steve Carr, and Philip Sweany: Global Register Partitioning. Proceedings of the 2000 International Conference on Parallel Architectures and Compilation Techniques, 2000.
17. Andrei Terechko, Erwan Le Thénaff, and Henk Corporaal: Cluster Assignment of Global Values for Clustered VLIW Processors. Proceedings of the 2003 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '03), 32–40, 2003.
18. Rainer Leupers: Instruction scheduling for clustered VLIW DSPs. Proceedings of the 9th International Conference on Parallel Architecture and Compilation Techniques (PACT 2000), 291–300, Oct. 2000.
19. Yi Qian, Steve Carr, and Philip Sweany: Optimizing Loop Performance for Clustered VLIW Architectures. Proceedings of the 11th International Conference on Parallel Architectures and Compilation Techniques (PACT 2002), Charlottesville, Virginia, Sep. 22–25, 2002.
20. Chung-Kai Chen, Ling-Hua Tseng, Shih-Chang Chen, Young-Jia Lin, Yi-Ping You, Chia-Han Lu, and Jenq-Kuen Lee: Enabling Compiler Flow for Embedded VLIW DSP Processors with Distributed Register Files. ACM SIGPLAN/SIGBED 2007 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'07), San Diego, Jun. 2007.