In Proceedings of the International Symposium on System-on-Chip, pages 137-140, Tampere, Finland, November 2006.

Register File Partitioning with Constraint Programming

Perttu Salmela, Chung-Ching Shen*, Shuvra S. Bhattacharyya*, and Jarmo Takala

Institute of Digital and Computer Systems
Tampere University of Technology
Tampere, Finland
{perttu.salmela, jarmo.takala}@tut.fi

*Department of Electrical and Computer Engineering and**
Institute for Advanced Computer Studies
University of Maryland, College Park, MD, USA
{ccshen, ssb}@eng.umd.edu

Abstract— **Highly parallel processors call for high bandwidth register access. One solution is to use multi-port register files. However, such register files are expensive in terms of chip area and their access time can lower the maximum clock frequency of the processor. Therefore, partitioning the multi-port register file to several smaller register files with less ports is preferred. In this paper, the partitioning of the register file is studied and the problem is formalized into such a form that constraint programming paradigm can be used to search for applicable register file partitionings. With the aid of the proposed method, applicable register file partitions with no performance penalty can be found and the inherent drawbacks of multi-port register file can be avoided.**

I. Introduction

Dividing the multi-port monolithic register file (RF) into several smaller RFs having only one or few read or write ports leads to significant savings both in chip area and power consumption and shortens the access time of the registers. Thus, partitioned RF is a preferred design choice for highly parallel processor architectures. For these reasons, in addition to register assignment, also the mapping of registers into separate RFs must be considered. The RF partitioning should not decrease the bandwidth of the register access and it should not require more RFs than necessary.

The problem of register assignment with single monolithic multi-port RF is a well studied area. The purpose of register assignment is to map variables or intermediate values of the program to registers. Typically, the assignment is expressed as a graph coloring problem [1][2]. However, application specific instruction set processors (ASIP) or very long instruction word processors (VLIW) can be capable of such an extensive parallelism that monolithic multi-port RF becomes too complex. With large number of ports, chip area grows linearly with the number of registers and quadratically with the number of ports [3]. RFs have also great effect on the energy consumption of the processor [4]. RFs must have less ports and, therefore, the processor architecture must possess limited connectivity [5]. Yet another possible optimizations are to divide RF in hierarchical manner according to the frequency of the register accesses [6] or to enhance RF partitioning with value cloning, i.e., the same value is kept in several RFs to enable prompt access to the value [7].

In this study, the register file partitioning for customizable highly parallel processors is considered. For example, the method can be applied on VLIW processors. It is assumed that the processor is tailored according to the application program. In this case, a suitable RF partitioning can be applied when the processor is generated. As an example case, the proposed method is applied on transport triggered architecture (TTA) processor [8][9]. On the contrary to register assignment techniques targeted directly for partitioned RFs [10], our approach is to have RF mapping as an extra step after the register assignment. The compiler can assume monolithic RF with unlimited number of ports which is convenient if a simple compiler is used. After the compiler has made register assignments, which minimizes the number of required registers, the mapping of symbolic registers to RFs is carried out. This mapping of registers to RFs is studied in this paper. Register assignment to multiple RFs in loops is studied in [10]. In our approach, the initial assignment of intermediate values to symbolic registers remains untouched. Thus, the proposed method is immune to the loops.

In the proposed approach, the core technique of solving the register to RF mapping is based on constraint programming (CP) [11]. Like in register assignment problem, we describe applying partitioned RFs also as a graph coloring problem. Partitioned RFs are targeted also in [12] where vertical and horizontal distribution enhanced with heuristics are applied and only a slight performance degradation is reported. Heuristics and graph presentation are also used in [13] but the graph has different semantics than in our problem description.

In the proposed method, the problem is described in such a way that the constraints of the RF partitioning can be taken into account in a convenient way. The method is exemplified with two realistic functions, the innermost loop of the fast Fourier transform (FFT) and the inner product. As a result, applicable RFs partitionings can be found and the required bandwidth of the register accesses is preserved. Due to the conflict free access, there is no performance degradation but all the obvious advantages of partitioned RFs are obtained.

II. RF Partitioning Problem Description

In order to fully exploit the available computation units in highly parallel processors, the operands must be fetched and

instr. #	read	write
0	$r1, r2$	$r4$
1	$r2, r3$	$r5$
2	$r4, r5, r7$	$r4, r5$
3	$r1, r6, r7$	$r4, r7$

a)

RF #	registers
0	$r5, r6$
1	$r1, r3, r4$
2	$r2, r7$

b)

Fig. 1. (a) Example of parallel register read and write accesses of four sequential instructions. (b) Mapping of registers $r1, \ldots, r7$ to RFs.

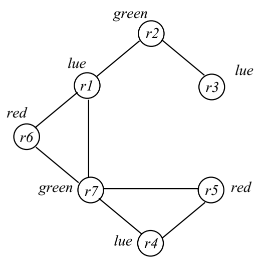

Fig. 2. Example problem as a colored graph. Connected vertices indicate simultaneously accessed registers. Colors could be used as labels of RFs.

the results passed without stalls. Therefore, multiple parallel register accesses are required. In Fig. 1(a) an example of accessing seven registers is shown.

The goal of RF partitioning in the example in Fig. 1 is to map registers $r1, \ldots, r7$ to RFs so that at maximum one read access and one write access per RF takes place during each of the instructions. It is assumed that the same register can be accessed simultaneously through the same port. Thus, the compiler is allowed to generate instructions like `add r1, r1, r1`. Both of these assumptions are satisfied with the targeted TTA processor. One of the possible solutions of the example problem is shown in Fig. 1(b). However, the complexity of the solution space increases rapidly with the problem size, and the problem is in general NP hard.

III. Partitioning with CP

Mapping of registers to RFs can be formalized as constraint satisfaction problem. It can be also noted that the problem without register file size restrictions can be defined as graph coloring problem.

A. RF Partitioning as Graph Coloring Problem

In graph coloring problem, a coloring is solved for a graph consisting of vertices V and edges E. The coloring must follow a rule such that for each vertex pair (V_i, V_j) connected by an edge $E_{i,j}$ the colors of the vertices V_i and V_j must be distinct. In other words, any of the neighboring vertices, i.e., vertices connected with single edge, must not have the same color. Naturally, there can be several feasible colorings and the number of used colors should be minimum.

The example problem in Fig. 1 can be expressed as a colored graph so that registers are described as vertices. Two vertices are connected with an edge if they are read simultaneously or written simultaneously. A colored graph of the previous example is presented in Fig. 2. In the figure, there is a one to one mapping of colors *red, blue,* and *green* to RFs 0, 1, and 2. For an arbitrary graph, the coloring problem is NP-hard. Due to the complexity, heuristic methods like, e.g., Brélaz's algorithm [14], can be applied for large problems. On the contrary to the presentation in Fig. 2, a graph where edges connect destination and source registers of atomic operations is used in [13].

B. CP Paradigm

CP is a technique for declarative description and solving of, especially combinatorial, problems. Therefore, it has numerous application areas and many of them in the operation research field. Constraint satisfaction problem is defined as:

- variable set $X = \{x_1, x_2, x_3, \ldots, x_n\}$,
- domains, i.e., finite sets of allowed values, D_i for each variable x_i, and
- constraint set, which limit the values that variables can have simultaneously. The constraint can be given as any expression which can be evaluated to Boolean value and whose argument is some set of assigned variables.

The values in domain can be any symbols and each variable can have a domain of its own. After the problem is described, a *solver* is used to find a feasible solution. The CP is not an optimization method since there is no cost function to minimize. In practice, the main programming effort of the CP is spent in describing the problem. The way how the problem is described effects heavily on the computation time of the solver. There exist many solving strategies [11]. In this work simple backtracking is used to search a feasible solution.

Backtracking solver assigns values to the variables one by one. After assignment, all the constraints, which are functions of variables assigned so far, are checked. If some of the constraint is violated, the most recently assigned value is changed to the next value in its domain. If all the values in domain are tried, the algorithm backtracks to the last assigned variable which still has alternative values to try. The algorithm does not assign values to all of the variables at once. Instead, it proceeds with partial assignment, which is extended in each forward step. In backtracking, assigned variable becomes unassigned until they get new values in the forward steps.

Naturally, the problem must be of moderate size due to the simple solving strategy. For efficiency, the constraints should be functions of as few variables as possible. A violated constraint of several variables does not indicate clearly enough which assignment has caused the violation. If the problem has no solution, whole multidimensional space of all the partially feasible solutions is traversed. Thus, the CP lends itself to the problems which have several solutions but it is enough to obtain only one feasible solution.

C. RF Partitioning as Constraint Satisfaction Problem

With the previous definition of the constraint satisfaction problem, the RF partitioning problem without RF size limitations can be defined with the variables, domains and constrains. The unknown variables are N registers, whose possible values are RFs. The number of RFs is N_{RFs}. The constraints are defined with the aid of r_i^{RF}, which indicates the RF mapped to the register r_i. Thus, the problem is defined with,

- variables $R = \{r_1, r_2, \ldots, r_N\}$,
- domain $RFD = \{0, 1, \ldots, N_{RFs} - 1\}$, and
- constraints $r_i^{RF} \neq r_j^{RF}$ for all pairs (i, j), $i \neq j$, where r_i and r_j are read during the same instruction or r_i and r_j are written during the same instruction.

On the contrary to the general problem description, the domain RFD is the same for each r_i. Although, with individual domains RFD_i some registers could be restricted to only certain set of RFs without extra efforts.

There is a clear analogy between the RF partitioning problem and the graph coloring problem in Fig. 2. The defined constraints correspond with the rules restricting the colors of connected vertices in graph coloring problem. The set of variables is analogous to the vertice set. RFs with size limitations can be also defined. Introducing a new constraint for RF utilization would be inefficient for backtracking. A constraint limiting RF utilization would be a function of every variable r_i and it could be evaluated only after all the variables are assigned. In case of violation, all the variables would be equally probable causes of the failure. Problem solving with such a constraint resembles systematic search by trying all the possible values. On the contrary, well defined constraints exclude parts of the solution space and all the possible value assignments are never tried.

RF size limitations can be taken account more efficiently by extending the domain to include both the RF and the physical register inside the RF. Although, extension of domain can increase the problem size so extremely that the solving would take infeasible long time. First, a set of physical registers in RF is defined as $RD = \{0, 1, \ldots, N_{RFsize} - 1\}$ where N_{RFsize} is the size of the RF. Next, the domain for register variables is defined as Cartesian product of RFs and physical registers inside the RFs. The physical register mapped to register variable r_i is denoted by r_i^R in constraint definitions. So, the problem can be defined as

- variables $R = \{r_1, r_2, \ldots, r_N\}$,
- domain $RFSZD = RFD \times RD$,
- constraints $r_i^{RF} \neq r_j^{RF} \vee r_i^R \neq r_j^R$ for all pairs (i, j), $i \neq j$, and
- constraints $r_i^{RF} \neq r_j^{RF}$ for all pairs (i, j), $i \neq j$, where r_i and r_j are read during the same instruction or r_i and r_j are written during the same instruction.

Yet more restricted RFs of unequal sizes could be defined by introducing individual sets of physical registers RD_j for each RF with domain $RFSZD = RFD \times RD_j$ where $j = 0, 1, \ldots, N_{RFs} - 1$. Furthermore, if register is allowed to be mapped only to certain RFs and the sizes of RFs are unequal the domains for each r_i could be defined as $RFSZD_i = RFD_i \times RD_j$ where $i = 0, 1, \ldots, N$ and $j = 0, 1, \ldots, N_{RFs} - 1$.

TABLE I
STRUCTURE AND RELATIVE COSTS OF MONOLITHIC RF FOR THE NON-PIPELINED AND PIPELINED ITERATIONS OF THE INNERMOST LOOP OF THE FFT AND FOR THE INNER PRODUCT.

		FFT	FFT pipelined	inner product
RF size	n	15	26	6
ports	p	4	6	3
RF connections / bus	$2p$	8	12	6
relative RF costs	$n(2p)^2$	960	3744	212

TABLE II
STRUCTURE AND RELATIVE COSTS OF PARTITIONED RFS.

		FFT	FFT pipelined	inner product
RFs	N_{RFs}	4	11	4
total RF size	$\sum n_i$	15	26	6
ports	p	1	1	1
RF connections / bus	$2N_{RFs}$	8	22	8
relative RF costs	$\sum n_i(2p)^2$	60	104	24

D. Application on FFT and Inner Product Functions

The proposed method is applied as follows. First, the C programs are compiled and scheduled to use monolithic RF. The first step generates parallel assembly code as a result. Next, the parallel assembly code is fed to a utility program that extracts parallel register read and write accesses of each instruction. This information is fed to the solver program which forms the problem, solves it, and proposes RF partitioning.

The RF partitioning method is applied to the innermost loop of the 1024-point radix-4 FFT and to the inner product. Both software pipelined and non-pipelined versions of the FFT are experimented. The applied partitioning uses the first presented RFD domain. The FFT in Fig. 3 uses monolithic RF and special function units (SFU) for complex addition and multiplication, address generation, and coefficient generation. The proposed method is applied to generate RF partitioning based on the parallel register accesses in the Fig. 3. Similar program flow of the inner product is shown in Fig. 4.

In Table I the structure and relative costs of monolithic RF are shown. Since there are equal number of read and write ports in the applied RFs, the number of ports p is doubled in the equation of relative costs and bus connections. The costs should be compared to the results in Table II where partitioned RFs are applied to the demonstrated functions. Even if the number of RFs is higher in Table II than the number of ports in Table I, i.e., there are more connections between the RFs and internal data buses of the processor, savings can be obtained. Furthermore, only the required connections are instantiated in the targeted TTA processors.

IV. CONCLUSIONS

Due to the inherent complexity of multi-port RFs a method for partitioning RF was proposed. The method was targeted as a back-end of the compilation such that the compiler can assume monolithic RF. Thereafter, the proposed method is

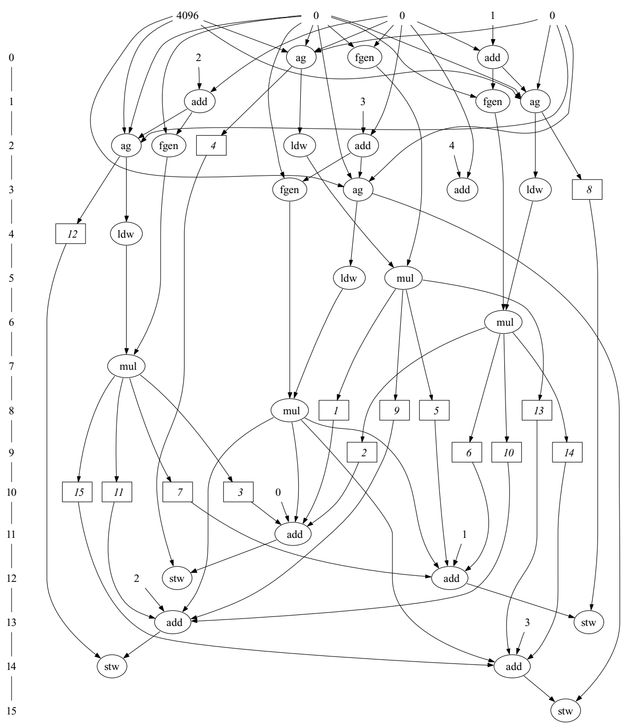

Fig. 3. Program flow of single non-pipelined iteration of the innermost loop of the FFT with monolithic RF. Processor contains the following SFUs, ag: address generation, cadd: complex addition, cmul: complex multiplication, fgen: coefficient generation. Due to the bypassing capability of the TTA processor some of the intermediate values are not passed via RF. Clock cycles are denoted on the left hand side.

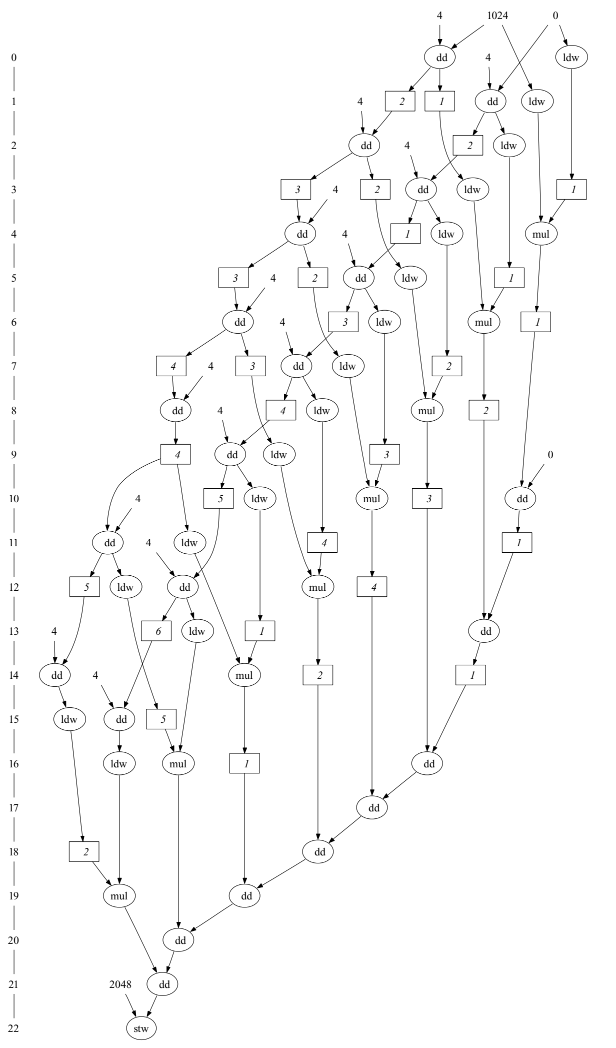

Fig. 4. Program flow of the inner product with monolithic RF.

used to re-map symbolic registers to separate RFs. The RF partitioning was shown to be equal with graph coloring problem. The formulation of constraint satisfaction problem was extended also to the RFs with size limits. The problem solving was based on backtracking CP algorithm. The applicability of the method was demonstrated with FFT and inner product functions. There was no performance degradation, since the method generates always conflict-free partitioning.

REFERENCES

[1] G. J. Chaitin, "Register allocation & spilling via graph coloring," in *SIGPLAN Symp. on Compiler Construction*, Boston, MA, USA, Jun. 1982, pp. 98–101.

[2] H. Krämer and W. Rosenstiel, "System synthesis using behavioural descriptions," in *European Design Automation Conf.*, Glasgow, UK, Mar. 1990, pp. 277–282.

[3] S. Rixner, W. Dally, B. Khailany, P. Mattson, U. Kapasi, and J. Owens, "Register organization for media processing," in *Intern. Symp. on High Performance Computer Architecture*, Toulouse, France, Jan. 2000.

[4] V. Zyuban and P. Kogge, "The energy complexity of register files," in *Int. Symp. on Low-Power Electronics and Design*, Monterey, CA, USA, Aug. 1998, pp. 305–310.

[5] A. Capitanio, N. Dutt, and A. Nicolau, "Partitioned register files for VLIWs: a preliminary analysis of tradeoffs," in *25th Ann. Intern. Symp. on Microarchitecture*, Portland, OR, USA, Nov. 1992, pp. 292–300.

[6] R. Balasubramonian, S. Dwarkadas, and D. H. Albonesi, "Reducing the complexity of the register file in dynamic superscalar processors," in *34th Ann. ACM/IEEE Inter. Symp. on Microarchitecture*, Austin, TX, USA, Dec. 2001, pp. 237–248.

[7] D. Kuras, S. Carr, and P. Swany, "Value cloning for architectures with partitioned register banks," in *Workshop on Compiler and Arch. Support for Embedded Systs.*, Washington D.C., USA, Dec. 1998.

[8] H. Corporaal, "Design of transport triggered architectures," in *4th Great Lakes Symp. on Design Autom. of High Perf. VLSI Syst.*, Notre Dame, IN, USA, Mar. 1994, pp. 130–135.

[9] ——, *Microprocessor Architectures from VLIW to TTA*. John Wiley & Sons Ltd, 1998.

[10] D. J. Kolson, A. Nicolau, N. D. Dutt, and K. Kennedy, "A method for register allocation to loops in multiple register file architectures." in *10th Intern. Parall. Processing Symp.*, Honolulu, HI, USA, Apr. 1996, pp. 28–33.

[11] R. Barták, "Constraint programming: In pursuit of the holy grail," in *Week of Doctoral Students*, Prague, Czech, Jun. 1999, pp. 555–564.

[12] J. Janssen and H. Corporaal, "Partitioned register file for TTAs," in *28th Ann. Intern. Symp. on Microarchitecture*, Ann Arbor, MI, USA, Dec. 1995, pp. 303–312.

[13] J. Hiser, S. Carr, P. Swany, and S. J. Beaty, "Register assignment for software pipelining with partitioned register banks," in *14th Int. Parall. and Distr. Processing Symp.*, Cancun, Mexico, May 2000, pp. 211–218.

[14] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, Apr. 1979.