IMPLEMENTATION OF A BAROTROPIC OPERATOR FOR OCEAN MODEL SIMULATION USING A RECONFIGURABLE MACHINE

Sayaka Shida, Yuichiro Shibata, Kiyoshi Oguri

Dept. of Computer and Information Sciences,
Nagasaki University, Japan
{shida,shibata,oguri}@pca.cis.nagasaki-u.ac.jp

Duncan A. Buell

Dept. of Computer Science and Engineering,
University of South Carolina, USA
buell@cse.sc.edu

ABSTRACT

This paper presents and discusses implementation of a barotropic operator used in ocean model simulation called Parallel Ocean Program (POP) using SRC-6 MAP. While a lot of high-end reconfigurable machines on which users can implement applications with a programming language are now available, enough implementation experience has not been accumulated for practical applications. In this paper, several implementation techniques accompanied by modification on original application source code are empirically evaluated and analyzed. The results show that appropriate use of internal memory and streaming DMA make 100 MHz FPGAs achieve comparative performance with GHz processors by using 100 MHz FPGAs.

1. INTRODUCTION

Reconfigurable devices such as field programmable gate arrays (FPGAs) have become an essential part of high performance general computing systems coupled with high-speed microprocessors. Many commercial high-end machines equipped with FPGAs are now available [1][2][3][4] and are used in performance-centric application fields including scientific simulation and cryptography [5][6][7].

While vast flexibility of FPGAs provides opportunities of efficient application implementation, there are also pitfalls that lead to inefficient implementation results[8]. In order to bring out the true value of reconfigurable machines, guiding principles of application design have to be established by accumulating implementation experience in a wide variety of practical application fields.

In this paper, we discuss implementation of a barotropic operator used in ocean model simulation called Parallel Ocean Program (POP)[9] on SRC-6 MAP[4]. While the compiler for SRC-6 translates software program description into FPGA configuration, users can explicitly specify a way of parallelization and data transfer in program description. Therefore, we will compare and analyze several implementation strategies in consideration of the architectural character of SRC-6.

2. SRC-6 ARCHITECTURE

SRC-6 is a high-end reconfigurable multiprocessor coupled with FPGAs as shown in Fig. 1. The SRC-6 is composed of Xeon 2.8 GHz microprocessors, reconfigurable processors called MAP, and Common Memory. They are connected each other with a crossbar switching system called Hi-Bar Switch. The MAP is composed of three Xilinx Virtex II XC2V6000 FPGAs.

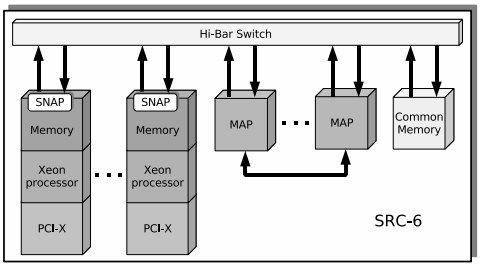

Fig. 1. Structure of the SRC-6.

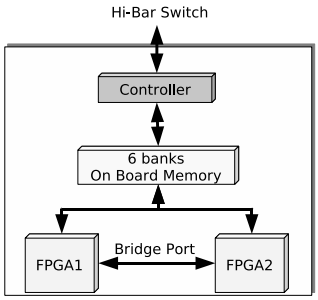

Fig. 2. Structure of MAP.

The structure of the MAP is shown in Fig. 2. Among the three FPGAs, one is dedicated to controller and the others (FPGA1 and FPGA2) are used for applications. The MAP has six banks of 4 MB On Board Memory (OBM), where data operated by FPGAs are stored. An FPGA can access multiple banks at the same time because they are connected with individual interactive ports. DMA controller also accesses the OBM so that the microprocessors and FPGAs can exchange data. The OBM is shared by FPGA1 and FPGA2. Arbitration of access to the OBM can be explicitly controlled by a user . The Bridge Port consisting of three interactive ports is also utilized to exchange data between FPGA1 and FPGA2.

1-4244-1060-6/07/$25.00 ©2007 IEEE.

```
Initial values AX(:,:,bid) = 0.0_8
do j=jb, je
  do i=ib, ie
    AX(i,j,bid) = A0 (i  ,j  ,bid) * X(i  ,j  ,bid) + &
                  AN (i  ,j  ,bid) * X(i  ,j+1,bid) + &
                  AN (i  ,j-1,bid) * X(i  ,j-1,bid) + &
                  AE (i  ,j  ,bid) * X(i+1,j  ,bid) + &
                  AE (i-1,j  ,bid) * X(i-1,j  ,bid) + &
                  ANE(i  ,j  ,bid) * X(i+1,j+1,bid) + &
                  ANE(i  ,j-1,bid) * X(i+1,j-1,bid) + &
                  ANE(i-1,j  ,bid) * X(i-1,j+1,bid) + &
                  ANE(i-1,j-1,bid) * X(i-1,j-1,bid)
  end do
end do
```

Fig. 3. Algorithm of the barotropic operator.

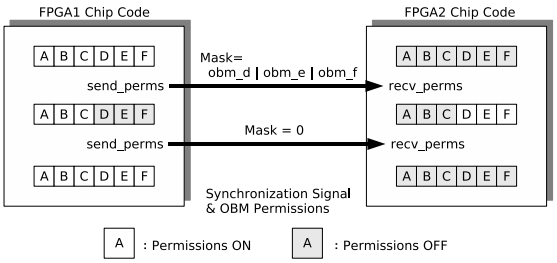

Fig. 4. OBM permission.

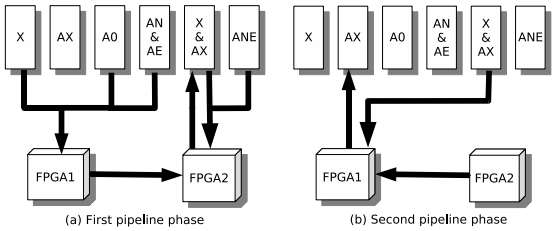

Fig. 5. Pipeline phases when using two FPGAs.

The SRC-6 has a unique compiler which generates a relocatable object file from C or FORTRAN source files. The compiler also translates the loops specified by users into hardware circuits to be configured on FPGAs. The compiler automatically analyzes structure of the loops and generates hardware that processes the repetition in a pipelined manner.

3. PARALLEL OCEAN PROGRAM

We took Parallel Ocean Program (POP)[9] as an example implementation target. The POP is an ocean circulation model derived from earlier models in which depth is used as the vertical coordinate. It solves three-dimensional primitive equations for fluid motions on the sphere under hydrostatic and Boussinesq approximations. Recently, the POP is used as a test application for high-end computers including reconfigurable machines [10][11]. The POP is also utilized as a basic model to build up a hybrid ocean circulation model and for weather simulation.

Our primary analysis using the GNU profiler revealed that the Preconditioned Conjugate Gradient (PCG) function was most time-consuming in all of the POP functions. This PCG function is used to solve the equation $Ax = b$, where the operator A is a matrix of nine-point stencils , and b is a vector of barotropic distribution. However, since the function is too large to fully implement on MAP, we implement a barotropic operator which is a dominant function of the PCG function. Fig. 3 shows the main algorithm of the barotropic operator function.

4. DESIGN STRATEGIES

The SRC compiler automatically generates hardware from user specified functions written in C or FORTRAN. At the same time, the compiler allows users to control structure of generated hardware like utilization of OBM banks and FPGA memories, and DMA styles and so forth. In order to efficiently implement applications, several design techniques are applicable which are accompanied by modification of source code.

4.1. Array Clipping

The barotropic operator function in the POP accesses a lot of large three-dimension arrays that are globally defined. Straightforward implementation with no code modification will need to transfer all these arrays into OBM with DMA, since FPGAs can not directly access to the host main memory where global data are located. In the barotropic operator function, however, only part of the arrays is accessed. Therefore, the arrays can be clipped so that only the required portion is transferred to reduce the transfer time.

As shown in Fig. 3, all of the arrays accessed in this function has the same third index *bid*. Therefore we modified source code to have two-dimension arrays.

4.2. Using two FPGAs

A process allocated to MAP may be distributed for the two FPGAs to achieve higher performance. Also, a large algorithm that can not fit in a single FPGA may be implemented using two FPGAs.

As described in Section 2, OBM can be used for communication between the two FPGAs, but one of the FPGAs has permission to access an OBM bank at the same time. By default, FPGA1 is allowed to access all of the OBM banks. Users can explicitly transfer the permission by calling system functions as shown in Fig. 4.

In our implementation, the arrays *A0*, *AN*, and *AE* in Fig. 3 are used by FPGA1, and *ANE* is processed by FPGA2. Array *X* is copied to two OBM banks, one for FPGA1 and the other for FPGA2. The barotropic operator function is implemented as two-phase pipeline structure as shown in Fig. 5. Fig. 5 (a) shows the first phase pipeline in which calculation is done by the two FPGAs In the second phase, the results of FPGA2 are sent to FPGA1 by changing the permission. The results of the two FPGAs are added and stored into the array *AX* as shown in Fig. 5 (b).

4.3. User macros

While the SRC compiler automatically generates circuits from high-level programming languages,we can also insert user macros written by Verilog or VHDL into source code.

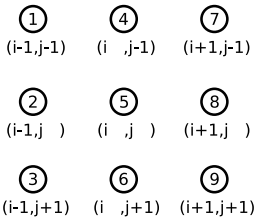

Fig. 6. Array *X* used by the barotropic operator.

In our implementation, a simple macro that contains a flip-flop is used to explicitly specify points of division of clock cycles. In this way, behavior of hardware to be generated can be easily controlled. However, since the operational frequency of SRC-6 is fixed to 100 MHz, it is important to pay attention not to drop the frequency.

4.4. Using Internal Memory

In the barotropic operator function the nine-point values (No. 1 to No. 9) in the array *X* shown in Fig. 6 are used in the same iteration of the main loop. In the next iteration, the values of No. 4 to No. 9 are used again. Since these values are stored on the OBM in straightforward implementation clock cycles per iteration of the pipeline hardware is increased due to the access latency to the OBM. Therefore, we used FPGA internal memory such as registers to alleviate the required access bandwidth to OBM.

In the target function, this method was applied for the arrays *AN*, *AE* and *X* shown in Fig. 3. Although the array *AN* was also a candidate of this method, it requires large internal memory space since all the elements of *AN* have loop-carried dependency in the outer loop. On average, almost half of the OBM access can be reduced by using FPGA internal memory whose latency is lower than that of OBM.

4.5. Streaming DMA

SRC-6 supports two styles of DMA transfer, regular DMA and streaming DMA. In the regular DMA, FPGAs have to wait for the finish of the transfer before starting the calculation. On the other hand, transfer and computation can be parallelized with the streaming DMA.

Fig. 7 shows the concept of the streaming DMA. In SRC-6, a pair of system functions is available for streaming DMA, stream_dma_cpu / get_stream. These functions create separated sections on the FPGA. In Section 1, data is continuously transferred through an OBM bank. In Section 2, data are fetched from the stream and used for calculation. These sections can be operated in parallel. The speed of streaming DMA is changed depending on the loop structure.

In our design, one of the arrays, *A*0, was transferred with streaming DMA. Since the array *A*0 is not reused, registers to store the streamed values are not required. Due to constrains of the DMA control mechanism, all of the arrays could not be transferred in the streaming style at the same time.

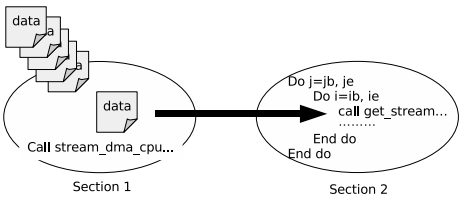

Fig. 7. Streaming DMA.

Table 1. DMA time.

	Execution time[ms]	DMA time[ms]
Full arrays	26.1	1507.5
Clipped arrays	26.1	63.3

5. EVALUATION

We implemented the barotropic operator function using the techniques described in the previous section and executed on the MAP of SRC-6. The evaluation results and also compared to software execution on Pentium 4 and Xeon processors.

First, we evaluated the effect of array clipping mentioned in Section 4.1 . As shown in Table 1, array clipping achieved 23.8 times speedup of DMA transfer of the arrays. On the other hand, execution time was almost the same, showing effectiveness of this kind of source code modification.

Then, we implemented POP in the four patterns of design architecture listed below. Based on the above mentioned results, array clipping is used on all of the architecture.

- Arch. 1 : Use of clipped arrays (Section 4.1) and a flip-flop macro (Section 4.3) on two FPGAs (Section 4.2)
- Arch. 2 : Use of internal memory (Section 4.4) in addition to Arch. 1
- Arch. 3 : Use of streaming DMA (Section 4.5) in addition to Arch. 1
- Arch. 4 : Use of internal memory and streaming DMA in addition to Arch. 1

In POP, the target domain of simulation is divided into multiple blocks, and an arbitrary block size can be chosen depending on architectural character of a simulation platform. Here, we chose and compare two block sizes, 16 and 64. When the block size is 16, the barotropic operator function requires 14 KB of data to be transferred by DMA. When the block size is 64, data to be transferred is increased to 224 KB. A large block size also increases the number of iterations in the barotropic operator function.

Fig. 8 shows the execution results when the block size is 16. The execution time of software processing on a 3.8 GHz Pentium 4 processor and a 2.8 GHz Xeon processor. DMA transfer time is a bottleneck for the SRC-6, and the Xeon processor shows the best performance in this case. Moreover, the execution times for Arch. 3 and Arch. 4 are longer

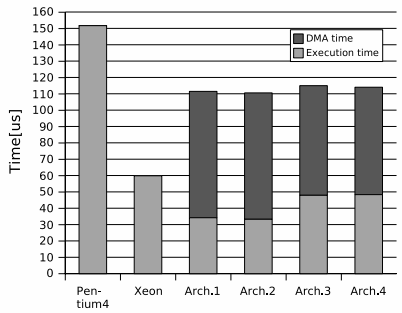

Fig. 8. Performance results (block size = 16).

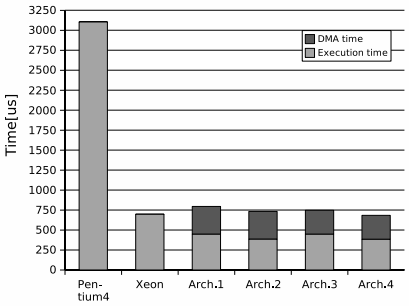

Fig. 9. Performance results (block size = 64).

than those for Arch. 1 and Arch. 2. This comes from relatively small data size of streaming DMA, which exposes initial overheads of transfer and idle time of the FPGAs. When the amount of data transfer is 14 KB, the bandwidth of DMA transfer is barely 0.18 MB/s.

Fig. 9 shows the results when the block size is 64. Compared to Pentium 4, Arch. 1, Arch. 2, Arch. 3 and Arch. 4 achieved 3.90, 4.23, 4.15 and 4.54 times speedup, respectively. While the use of internal memory was faster than streaming DMA, the best performance was marked by using both of the techniques at the same time. The larger the block size becomes, the smaller portion the DMA overhead makes up. Even the fastest Arch. 4 marked only 1.02 times speedup compared to the Xeon processor, while the improvement of execution time stood at 1.82 times. The results again suggest the importance of the implementation techniques including source code modification.

Execution time for the FPGAs would be further improved by appropriately allocating data on OBM banks to allow multiple access. However, limitation of FPGA size prevents this since a wider bandwidth of OBM introduces further parallelism and thus more arithmetic modules. The device utilization when the block size is 64 is shown in Table 2. While about 40% of FPGA2 slices are utilized, the slice utilization ratio for FPGA1 is around 60%. Consequently, current design is not able to increase the parallelism to achieve further performance improvement.

The MAP with the XC2V6000 FPGAs we used here is not a brand new model. The current MAP model features two XC2VP100 FPGAs and these larger FPGAs would alleviate the size limitation to some extent by doubling the parallelism of the barotropic operator function on FPGAs.

Table 2. Device utilization.

	Slices[%]		BRAMs[%]		MULTs[%]	
Design	FPGA1	FPGA2	FPGA1	FPGA2	FPGA1	FPGA2
Arch. 1	61	32	11	0	55	44
Arch. 2	62	37	11	0	55	44
Arch. 3	62	32	11	0	55	44
Arch. 4	65	37	11	0	55	44

6. CONCLUSION AND FUTURE WORK

In this paper, implementation of the barotropic operator in the POP on the SRC-6 is discussed. In consideration of the architectural character , several design strategies are compared. The results show that proper source code modification effectively reduces both of the execution and data transfer time, while the FPGA size mainly restricts the performance. In order to implement the full process of the ocean model simulation, dynamic reconfiguration of the FPGAs is required. Fortunately, the behavior of the ocean model simulator is rather regular and is easily predictable. Appropriate reconfiguration scheduling that makes reconfiguration overlapped with execution on the microprocessor will be possible. Evaluation of parallelization effects using multiple processors and MAP modules is also our future work.

Acknowledgment The authors would like to their express sincere gratitude to Dr. Saha of George Washington University for valuable advice.

7. REFERENCES

[1] "Cray Inc." http://www.cray.com/.

[2] "Silicon Graphics, Inc." http://www.sgi.com/.

[3] M. Gokhale, W. Holmes, A. Kopser, S. Lucas, and D. Lopresti, "Building and using a highly parallel programmable logic array," *IEEE Computer*, vol. 24, no. 1, pp. 88–89, 1991.

[4] "SRC Computers, Inc." http://www.srccomp.com/.

[5] K. Gaj, T. El-Ghazawi, A. Michalski, M. Huang, C. Shu, E. El-Araby, M. Taher, and P. Sha, "Recomfigurable Computing Machines: An emprical analysis," *SuperComputing*, 2003.

[6] R. Scrofano, M. Gokhale, F. Trouw, and V. K. Prasanna, "A hardware/software approach to molecular dynamics on reconfigurable computers," *Proc. FCCM*, pp. 23–34, 2006.

[7] M. C. Smith, J. S. Vetter, and S. R. Alam, "Scientific computing beyond CPUs: FPGA implementations of common scientific kernels," *Proc. MAPLD*, 2005.

[8] O. Mencer, "Computing with FPGAs," *Proc. COOL Chips*, pp. 327–343, 2006.

[9] "Los Alamos Climate Ocean and Sea Ice Modeling:models:POP," http://climate.lanl.gov/Models/POP/.

[10] P. H. Worley and J. Lecesque, "The performance evolution of the Parallel Ocean Program on the Cray X1," *Cray User Group Conference*, 2004.

[11] M. R. Fahey, S. Alam, J. Thomas H. Dunigan, J. S. Vetter, and P. H. Worley, "Early evaluation of the Cray XD1," *Cray User Group Conference*, 2005.