Automatic Partitioner for Behavior Level Distributed Logic Simulation

Kai-Hui Chang[1], Jeh-Yen Kang[1], Han-Wei Wang[2], Wei-Ting Tu[2], Yi-Jong Yeh[1], and Sy-Yen Kuo[2]

[1] Avery Design Systems, Inc., Andover, MA 01810, USA
[2] Graduate Institute of Electronics Engineering, National Taiwan University, 1, Sec. 4, Roosevelt Rd., Taipei, Taiwan 106, ROC
sykuo@cc.ee.ntu.edu.tw

Abstract. As the complexity of circuit design increases, verification through simulation has become a bottleneck of the IC design process. Distributed parallel simulation is one way to solving the problem. In order to distribute the simulation workload to multiple processors, the design must be carefully partitioned first. While most previous work focus on gate level partitioning, our work extends a previously implemented Verilog gate-level partitioner to support RTL and behavior level partitioning. Techniques to partition special constructs specific to these levels, such as global access, function calls and memory access, are described in this paper. The experimental results show that our techniques are capable of finding partitions which can accelerate simulation.

Keywords: distributed simulation, parallel simulation, RTL level partitioner, behavior level partitioner.

1 Introduction

As the complexity of circuit design increases, verification through simulation has become a bottleneck of chip design process. Large designs, such as System on Chip (SOP), may take several days to simulate. Sitting for simulation to finish is a waste of time and increases the chip's time to market. Therefore the Field Programmable Gate Array(FPGA) prototyping, such as an emulator, has been proposed to solve this problem. However, emulators are expensive and hard to use.

Memory usage is another problem that arises when a design gets larger. The physical memory of a computer is limited, so it may not be possible to simulate large designs that exceed the computer's capacity of memory.

Several Electronic Design Automation (EDA) tool vendors now provide distributed simulation solutions to solve the problem, e.g. the Simcluster [1]. Through distributed simulation, the workload can be distributed among several processors and improve the turnaround time of simulation. By dividing a design into several smaller pieces, each piece needs less memory for simulation

F. Wang (Ed.): FORTE 2005, LNCS 3731, pp. 525–528, 2005.

and can easily fit into a modern computer. However, for better performance, partitioning must be done carefully.

Partitioning has been studied for several decades and many algorithms have been proposed and solved the partitioning problems successfully. However, most of them focus on flattened gate-level partitioning. As chips get larger, RTL and behavior level simulation have also become bottlenecks in the IC design process, and the demand for parallel simulation is increasing. Furthermore, in a large project, it is often necessary to simulate a design containing several levels of abstraction. For example, part of a chip may be in gate-level while another part in RTL-level; or part of the design may be hierarchical while the other part is flattened. This emerging demand for mixed abstraction level of parallel logic simulation is beyond the capability of existing gate-level partitioning algorithms.

In this paper, a partitioner that can perform the partitioning at all abstraction levels is proposed. It can find partitions for a design from behavior level to gate level, no matter it is hierarchical or flattened.

2 Previous Work

The goal of the distributed simulation partitioner is to make the workload shared among processors as balanced as possible and make the communication overhead as small as possible. A good survey on this topic is given by Bailey, Briner and Chamberlain [2]. While most previous work focuses on gate-level, Guettaf et al. [3] proposed a behavior level partitioner for VHDL. However, they did not address how to handle special constructs like function call, global access and memory access, and these issues are addressed in our work.

The multi-level partitioner proposed in this paper is relied on our previous work on gate-level partitioning. It supports two partitioner modes, *normal mode* and *regroup mode*, and utilizes techniques to flatten the design for finer-grained partitioning. Please refer to [4] for more details on this work.

3 Behavior and RTL Level Partitioning

Our proposed workload estimation technique and RTL/behavior level partitioning algorithm are discussed in this section.

3.1 Workload Estimation

Workload in a simulation consists of processing signal changes, scheduling the affected behaviors, and executing the affected behaviors. In general, it is difficult to estimate workload accurately, and approximations are needed. In gate level designs, the number of gates are often used to estimate workload in each partition. In this paper, we use the number of variables and nets in a partition to estimate the workload, which is similar to the use of gate count in gate-level designs.

3.2 Partition Algorithm

The gate-level partitioner we based on is only capable of partitioning gates (or instances in Verilog). In order to handle RTL/behavior level code, we need to convert them to instances first, and the algorithm is given below.

For each *initial block, always block*, and *continuous assignments*, an instance is created. The variables used at the LHS(left hand side) of the assignments will become output ports, and the variables used at the RHS of the assignments will become input ports. Then it can be partitioned with any gate-level partitioning algorithm.

Task and *function* definitions are duplicated to all the partitions where they are called. Since Verilog *task* and *function* calls are not reentrant, this approach will not cause any problem.

Global accesses which cross partitions will be replaced by *channel commands* provided by Simcluster. In Simcluster, *channel commands* are provided to set, view, force and release variables in a remote partition. For example, "a= b.c.d;" will be converted to "$channel_variable("b.c.d", a)".

Memory accesses are handled differently from variables because a memory cannot become a port. Task and function are used to handle memory accesses that cross partitions, and the template is given in Figure 1. In the template,

When memory access to "a.b.mem" appears on the LHS and is in another partition:

```
a.b.mem[i]= RHS;
```

will become:

```
set_mem(i, RHS);

task set_mem;
input index;
input RHS;
$channel_memory_set("a.b.mem", index, RHS);
endtask
```

When memory access to "a.b.mem" appears on the RHS and is in another partition:

```
LHS= a.b.mem[i];
```

will become:

```
LHS= fetch_mem(i);

function fetch_mem;
input index;
reg result;
begin
$channel_memory("a.b.mem", index, result);
fetch_mem= result;
end
endfunction
```

Fig. 1. Remote Memory Access Template

$channel_memory is used to get data from the memory in a remote partition, and *$channel_memory_set* is used to send data to the remote memory.

4 Experimental Results

The design is part of the CF FFT project from Opencores [5]. It is a fast Fourier transform converter. The FFT architecture is pipelined on a rank basis; each rank has its own butterfly and ranks are isolated from each other using memory interleavers. The design is in RTL level. The "large testbench" provided in the project is used as the testbench and is also in RTL level. The cf_fft_4096_16 configuration (4K point FFT, 16 bit precision) is used as the design under test. The top-level module instantiates the testbench and the design under test. Our partitioner is used to partition the design into two partitions at top module.

The simulator used is VCK, and the platform is Redhat Linux 8.0 running on a workstation with dual AMD MP 1.8GHz CPU. The partitioner run time is 0.6 seconds. The single process run time is 118 seconds. After partitioning, the distributed run time becomes 71 seconds with 1.66X speed up.

Aside from this benchmark, our partitioner has also been used by Avery Design Systems to partition several commercial designs successfully.

5 Conclusion

In this paper, we extended a gate-level partitioner to support RTL and behavior level partitioning. We proposed techniques to convert RTL/behavior level code to gates and described how to handle constructs specific to these levels. Unlike other partitioners, which usually focus on gate-level circuits only, our partitioner can partition any design in any level of abstraction, as long as it is written in Verilog. From the results of the experiment, it can be concluded that the partitioner proposed in our work can indeed find good partitions that accelerate circuit simulation by distributed simulation. With this partitioner, distributed simulation tools will become easier to use and greatly save circuit designers' and verifiers' time.

References

1. Avery Design Systems, "VCK/Simlib User's Guide", Rev. 1.1.0, May 2002
2. M. L. Bailey, J. V. Briner, Jr., and R. D. Chamberlain, "Parallel logic simulation of VLSI systems", *ACM Computing Surveys*(1994)
3. A. Guettaf and P. Bazargan-Sabet, "Efficient Partitioning Method For Distributed Logic Simulation of VLSI Circuits", *Annual Simulation Symposium*, 1998, pp. 196-201.
4. K. H. Chang, H. W. Wang, Y. J. Yeh, and S. Y. Kuo, "Automatic Partitioner for Distributed Parallel Logic Simulation", *IASTED International Conference on Modeling, Simulation and Optimization(MSO'03)*, Kauai, Hawaii, USA, 2004.
5. CF FFT Project, http://www.opencores.org/