IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 6, Issue 6, Ver. I (Nov. - Dec. 2016), PP 01-04
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Partition and Sessionid Based Compilation Approach to Improve Re-Compilation Time and Efficient Use of Resources in Industry Standard Tools

Sachin Puranik[1]
[1]*Verification Engineer, ARM Inc, Bagmane World Technology Centre, Bangalore-560037*

Abstract: *The Design and Verification of SoCs is complex process due to multi-million transistors and the complexity of the actual design. The paper describes partition based compilation approach tagged with sessionids which can be derived to any industry standard VLSI design simulation tools. Thus if this methodology if implemented can be used to significantly reduce the compile time and manage the resources efficiently during re-compilations*

Keywords: *Partition, Compile, VCS, IRUN, VLSI, Modelling, recompilation*

I. Introduction

As we know these days the number of transistors in anSoC is increasing exponentially. Just as the great Gordon Moore proposed; which is now known as the "Moore's Law": "The number of transistors on an integrated circuit will double every two years". In the competitive VLSI markets such as Multimedia, Network and Mobile processors the time from market analysis to the actual Tape out time of anSoC are very stringent. The actual design and verification of anSoC eats up most of the time in the aforementioned design cycle of an SoC. The verification of the complex SoC consists of broadly two aspects viz. compiling the design with a testbench and simulate the design with a test vector and analyze the behavior is as expected. In the multimillion gate SoCs compile time of a RTL design can take half an hour to several hours and Gate level design can take from several hours to half a day. With the partition based compile approach along with a concept called sessionid the compile time can be improved drastically.

II. Partition and sessionid based compilation

A. Introduction to partition based compilation

Intuitive in the first glance itself word "partition" involves dividing the design into multiple logical submodules. For example let's take an SoC which has say a Processor, an Interconnect, a DDR; Interconnect also hosts other peripherals such as Sata,USB and say an I2C module as described in the Figure 1.

The actual verification of an SoC has to involve verifying the subsystem individually as well as functionality of the whole SoC.For example checking the functional correctness of SATA,USB and I2C as well as checking if the whole SoC is behaving as per the requirement. Since each subsystem itself hosts close to a million transistors, the SoC can have more than million gates.

So the industry standard tools now host a feature called partition based compilation ; ie each subsystem can be compiled individually on a separate machines and once the objects are ready they can be linked together to create a SoC executable database.

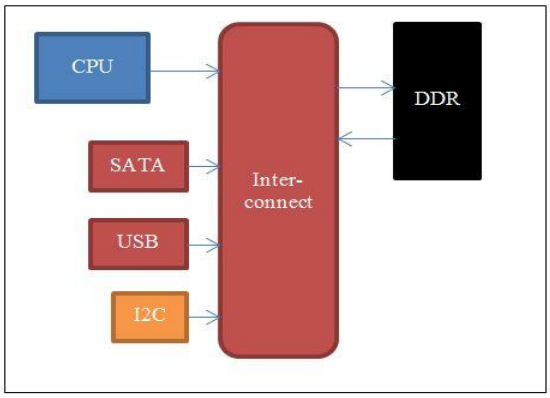

Figure 1: A sample SoC diagram

In order to prepare a design for a partition based compilation flow, one must first perform Analysis of the design and elaboration in order to identify the logical entities in the design. Once a design entity is set as a partition, hierarchical entities below are also assigned to that partition. By default, all the compilation software designates the top-level entity as a design partition. So one planning to use this technique must keep in mind that the logical partition that you are going to compile separately must not refer to resources from other logical partitions in the design in order to avoid the resource conflicts between partitions.

Figure 2 below describes a sample flow of how partition based compilation can be beneficial. As you can clearly see from the diagram any changes in the any of the logical partitions of the design doesn't require you to recompile the entire design which can reduce the compile time drastically.

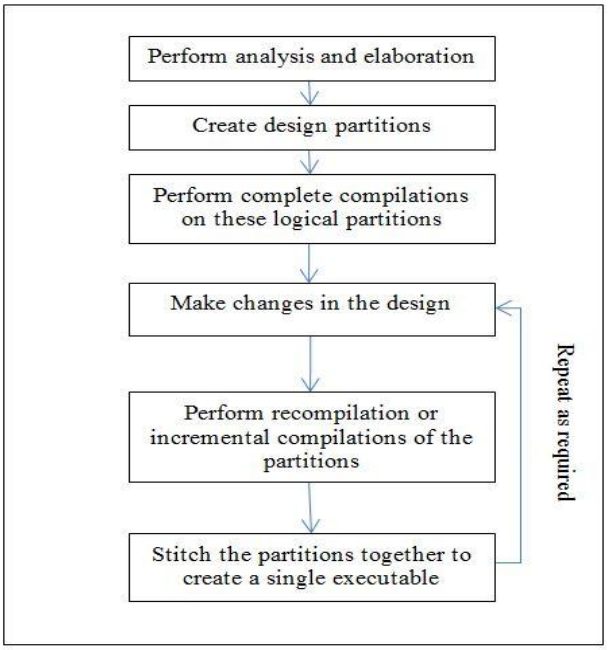

Figure 2: Flowchart explaining the steps involved in partition based compilation

Table 1 is a table comparing the actual compilation times without and with partitions. As we can clearly see the first fresh run takes approx. same time if run on the single thread machine. We can optimize the partition approach further by using different machines. Second and third runs clearly beats the normal approach by time reduction close to 50%.

	1st Run	*2nd Run without change*	*3rd Run with change in one of the partitions*
Normal approach	15:41 min	12:13 min	13:45 min
Partition approach	16 min	6:20 min	7:47 min

Table 1: A table indicating the time improvement

Achieved from the partition based compilation; run on a single thread machine

B. Introduction to sessionid based compilation

Sessionid technique can be extremely useful if we want to have single area with multiple databases for multiple test vectors. Consider figure 1 again; let's say in our SoC USB supports both host and device modes. If we want to test both these features on a new updated design simultaneously we will have to create two databases for the same copy of the design and then go on to create two snapshots; which consumes disk space that's double than the one that we would be using if we were to employ sessionid approach.

In this approach we initially download the new design; then in a single database we launch analysis, elaboration and compilation for USB host with a sessionid. In parallel we will start another compilation for USB device with different session id. Thus we can have single database with two different executables for different test vectors. Table 2 below explains how the sessionid approach helps in efficient utilization of storage space considering we have created two executables. 1GB reduction might seem too less but can be significant considering the current SoCs with 1000+ test vectors. It can be extremely helpful in case of Gate level simulations where a single design can consume 30+GBs and the design can be from 10+ GBs.

	Without sessionid	*With sessionid*
Disk utilization	8GB	7GB

Table 2: Resource consumption with and without session id

C. Sessionid tagged with partition flow.

Since these two methods help reducing the time needed for compilation and helps in reduction of resource utilization, I thought why not harness good from each method and employ it for the best compilation strategy?

The exploit that I used here is, we first compile the entire design with all permutations first based on partition approach. Mark it as reference design, Use this reference design and derive the required subset design based on changes to specific partitions using sessionids in the same database. As we can see from figure 3 once the reference design is compiled which is one-off and can take quite a bit of time we can derive as many subsets as we want based on the sessionid and required changes in the partitions.

Key thing here is in a session id based compilation if the partition from which is derived from is unchanged then it will just refer to that partition else it will create a separate partition which will consume storage.

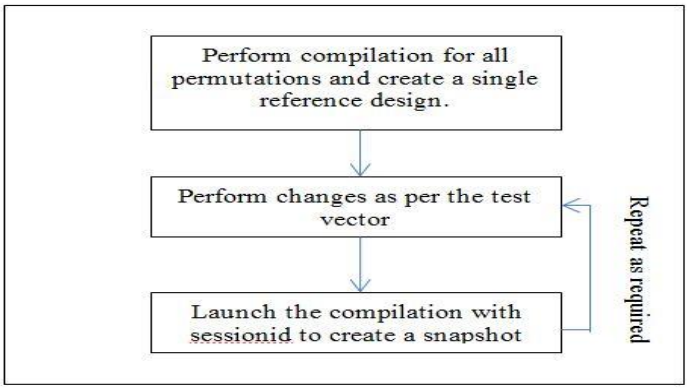

Figure 3: Flowchart explaining how to employ partition and sessionid approach

The following table 3,4 describe the results of the experimentation on different designs from the one above that I performed employing the sessionid and partition strategies. As we can clearly see by using the both strategies we can significantly improve the compile time and resource consumption.

	Reference compilation	*1st Run with change*	*Total disk usage*
Normal approach	29:13 Mins	25:10 Mins	10GB(need two databased)
Partition,sessionid approach	35 Mins	10:20 Mins	7GB(1 database one normal 1 with a session)

Table 3: Results from partition and sessionid approach in RTL compilations

	Reference compilation	*1st Run with change*	*Total disk usage*
Normal approach	3 Hrs 30 Mins	2 Hrs 15 Mins	60GB(need two databased)
Partition,sessionid approach	4 Hrs 00 Mins	44:76 mins	39GB(1 database one normal,1 with a session partitions)

Table 4: Results from partition and sessionid approach in Gate level compilations

III. Conclusion and Future Work

The verification cycle of any VLSI SoC is affected by the high compilation, simulation time and resource requirements. If we can employ the sessionid and partition approaches we can significantly optimize compile time and disk storage needed and can invest these resources in doing other important tasks.

The paper lays a promising foundation for the improvement of the any design compilation time, but in addition may be used to further ease up the compilation process. For example, while the IP deliveries are done they can deliver precompiled databases; all we have to do is just create reference pointers to those databases while creating the executable. However, resolving this is left as future work.

References

[1]. Synopsys VCS user guide version.
[2]. http://users.ece.utexas.edu/~patt/10s.382N/handouts/vcs.pdf
[3]. System- on -chip basics Springer-"Embedded Software Design and Programming of Multiprocessor System-onChip", Embedded Systems, Springer Science+ Business Media

Author Profile

Sachin Puranik received the B.E. degree in Electrical Engineering from R.V. College Of Engineering in 2013. During his tenure as a student in RVCE, he has contributed in Antenna related projects under the guidance of Mr.Ravishankar S, MrShushrutha; Professor and Associate Professors at RVCE. Then he went on to work with Freescale semiconductors in the field of Digital networking SoCs verification. He is now with ARM Inc. working on Memory management chip architecture verification