Expedite any Simulation with DMTCP and Save Decades of Computation

Balaji R, CAD Er, Intel Corporation, Bengaluru, India (*balaji.r@intel.com*)

Sathish Kumar Sugumaran, SOC Design Er, Intel Corporation, Bengaluru, India (*sathish.kumar.sugumaran@intel.com*)

Rohan Garg, Northeastern University, Boston, USA (rohgarg@ccs.neu.edu)

Jiajun Cao, Northeastern University, Boston, USA (jiajun@ccs.neu.edu)

Gene Cooperman, Northeastern University, Boston, USA (gene@ccs.neu.edu)

***Abstract*—The SoC simulation models are becoming more complex day by day, and therefore require more and more time to complete execution. Some models can take up to a week in order to complete a single simulation. Considering the general "shift left" testing strategy, it becomes increasingly important to run pre-silicon simulations efficiently in order to capture the design and content bugs early on, and reduce the number of steps required to achieve production readiness. This desire to reduce the number of steps, and therefore reduce the turnaround time, imposes several unique challenges. In either case, the simulation should be rerun from the beginning to resolve/triage the failure cases, which wastes many man hours and compute resources. This badly impacts the time-to-market (TTM), and eventually decreases revenue. To address these issues, a transparent checkpoint-restore capability using DMTCP (Distributed Multithreaded CheckPointing) [1] has been developed. The proposed approach improves the usage of compute resources by allowing for the bypass of initialization flows, such as full long reset flows that are common across the regressions. This also helps debug simulation failures by reducing the effective turnaround time. We collaborated with the Northeastern University team to improve the DMTCP checkpoint and restore solution for the simulation use cases. The DMTCP-based approach brings various advantages over the earlier BLCR approach [3]. This simulation checkpointing technique can greatly shorten the development and debug cycle in the Pre-Si environments. Depending on the test environment and test, if reset takes most of the wall clock time, then one can gain significant savings.**

Keywords—Save & Restore; Checkpointing; Distributed MultiThreaded CheckPointing; DMTCP; Simulation; Engineering Computing; Throughput; Turn-around time;

I. INTRODUCTION

As the semiconductor industry becomes more focused towards the "shift left" paradigm, the need for robust and timely validation of the design in Pre-Si becomes absolutely crucial. Simulation is still the work horse of Pre-Si validation. Its model availability is earlier in the PLC (Product Life Cycle) compared to other Pre-Si validation platforms like Virtual-Platform (VP), Emulation, and FPGA or GLS. Thus, any increase in bugs caught during simulation reduces the cost of the chip. When bugs escape to further down in the PLC, the cost of fixing them increases. Hence, in order to catch more bugs in the simulation, we need to come up with an efficient method to minimize the simulation time. Both environmental instability and an actual bug in the model force the user to execute more simulation reruns. The additional simulation reruns cause much time to be lost.

Post-Si debug would be straightforward if the validation content were error-free. However, this can only be achieved by running through the validation content exhaustively, which is prohibitively expensive. Instead, all execution teams should be able to take snapshots of the simulation run either at periodic intervals or when some particular event of interest occurs. By sharing these snapshots, the teams can cut the number of reset simulations.

The DMTCP save-restore technique provides validation engineers with the capability to do checkpoint and restore of the simulation run at the process level. DMTCP has the capability to checkpoint and restore multiple processes of an application. DMTCP also provides wrappers that abstracts out the high-level application. This avoids burdening the validation engineer with low-level changes between checkpoint and restore such as changing process id (pid), network sockets, file paths associated with open file handles, etc.

There were two other tools that were evaluated prior to this work: Berkley Lab Checkpoint/Restart (BLCR) [2] and the native save-restore functionality of the simulator. However, each of these tools had limitations. BLCR requires installation of kernel modules, and hence a mini-lab setup is required to get root access for this activity. The native save-restore of the simulator does not have support for multi-threaded process checkpointing, which is a common occurrence in complex simulation test benches/models.

The following table does a comparative analysis of features supported by DMTCP compared to BLCR. A fully implemented feature scores 1; a partially implemented feature with limitation scores 0.5; and an unimplemented or unavailable features scores 0. The details are included in reference [3].

Table I. DMTCP vs BLCR (see [3] for the original table)

Feature	DMTCP	BLCR
Workload Agnostic	0.5	1
Support for perl scripts	1	0
Support for interactive sessions (ttys, vnc)	1	0
Support for process groups and sessions	1	0
IPC (pipes) support	1	0
Support for network resource usage	1	0
Support for device files and /proc use	1	0.5
Support for multi-threaded processes	1	1
Support for distributed processes	1	0.5
Non-kernel dependent (User Space)	1	0
Process ID Virtualization	1	0
Checkpoint Image Compression	1	0
Extensibility - ability to write extensions	1	1
Final score	**12.5**	**4**

II. DMTCP

A. *DMTCP Overview*

DMTCP is a tool to transparently checkpoint the state of multiple simultaneous applications to disk, including multi-threaded and distributed applications. It works under Linux, with no modifications to the Linux kernel or to the application binaries. It can be used by unprivileged users; no root privilege is required. One can later restart from a checkpoint, or even migrate the processes by moving the checkpoint files to another host prior to restarting. A DMTCP checkpoint image includes any libraries (.so files) that it may have been using. This strategy is used for greater portability of the checkpoint images. In some cases, this allows for migration of checkpoint images to a host with a different Linux distribution and a different Linux kernel.

B. *DMTCP Advantages & Limitations*

DMTCP supports checkpoint-restore of distributed processes of a program. The processes may be connected through network sockets or through other inter-process communication mechanism such as POSIX shared-memory areas. This is a key advantage compared to other solutions and expands the scope of how this kind of save-restore technique can be used. It works completely in user space and doesn't require any modifications to the kernel, and hence it allows non-privileged users to use the solution. It is also open source and follows a plugin architecture, and hence it is easily extensible. It is tried and tested, and has been extremely stable in our experience.

However, DMTCP depends on standard symbol lookup mechanism provided by the dynamic linker-loader to achieve its goals. This works for most common use cases. However, in some unusual cases, such as a non-standard symbol lookup mechanism or system calls using inline assembly, DMTCP cannot track the system calls made by the process to the kernel. This was observed with signal dumper libraries where DMTCP was not able to track the system calls and the file handles opened by those libraries. It also cannot utilize the simulator internal mechanisms such as the save-restore functionality of the native simulator. For example, the save-restore functionality allows one to change command-line arguments after a restore, but it's not possible to directly use this mechanism with DMTCP.

III. USE CASES

A. Skipping Reset

With the help of event-based checkpointing, one can checkpoint the model design, based on different reset phases. This allows one to create as many checkpoints as the number of different phases in a single regression. This is illustrated in Figure I. Further, if the regression fails at any point of time, one can restore it from a previous checkpoint. This increases productivity by avoiding the redundant runs for every test.

The efficiency can be increased further by reusing the same checkpoint state across simulation runs. This is achieved by allowing users to change the sequence executed in a later test phase after restore. The reset phase can take up to 95% of the execution time and on average 50% to 60% of the execution time. This technique can be used to skip the reset phase and directly start executing the test phase.

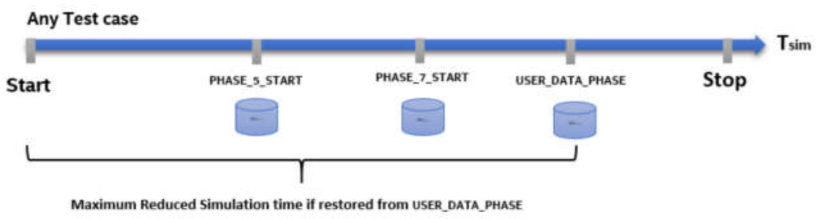

Figure I. Phase based checkpoint restore

For example, in Figure I, a test is snapshotted at PHASE_7_END and restored from the same point. This helps the different tests to skip the complete reset flow by changing the test sequence plugged in at USER_DATA_PHASE and start running the regressions right away in the restored run.

B. Debug Speedup

Periodic checkpoints can be used to improve the stability of a simulation run by taking a snapshot periodically and saving it to the disk. When the simulation fails due to a machine failure, it can be restored from the last checkpoint, instead of rerunning it from the beginning. Furthermore, if the failure is due to a model bug, checkpoint-restore can enable further debugging during restore by enabling signal dumps or by forcing the release or deposit of signals. This allows the user to quickly find the root cause of the issue, instead of rerunning the simulation from the start. This methodology is illustrated in Figures II and III.

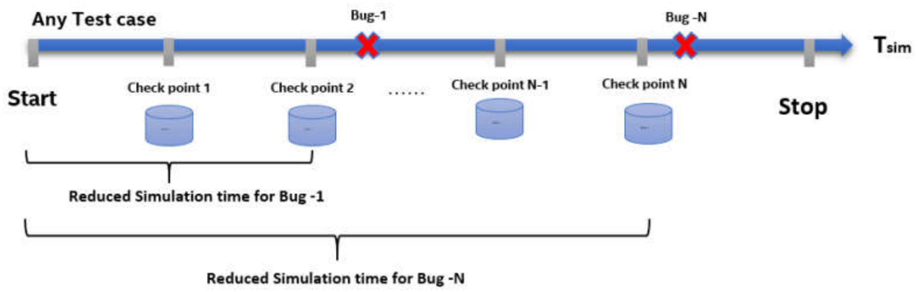

Figure II. Dynamic checkpointing based on clock cycles

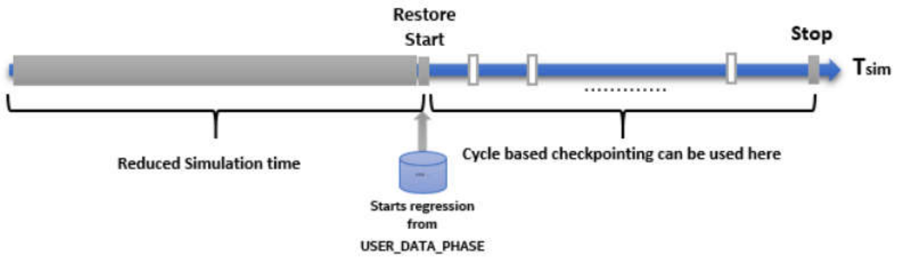
Restore
Start
Stop
Tsim
Reduced Simulation time
Cycle based checkpointing can be used here
Starts regression from USER_DATA_PHASE

Figure III. Efficient usage of event & cycle based checkpointing

C. Case Study with SCAN

SCAN validations require changing the SCAN pattern for different tests, while the simulation binary remains the same. Thus, Pre-Si SCAN validation can greatly benefit from save-restore by skipping the reset phase. For example, a typical validation use case with 'n' partitions, 'm' flavors of patterns, and 'l' iterations requires n*m*l repeated simulations. This translates into months of repeated reset simulations consuming many man hours. Most of the pattern validations require a similar reset state and are fixed well in advance of the reset usage for every team. There are very few cases where one needs different reset phases. This is treated as a separate case, but the same methodology is used. For the typical case, checkpoint-restore enables running of the 'n' different partitions and the 'm' different patterns in parallel using one saved state. This leads to only one parallel simulation for all of the m*n*l possible cases. In the case of a failed pattern, one just needs to rerun only the failed patterns. Note that the failed patterns can be run in parallel from a common saved state. Furthermore, the reset phase doesn't need to be executed for any of the parallel runs and the execution can jump directly to the pattern. This is the key advantage of the proposed approach: to save years of cumulative simulation time and many man hours, and eventually millions of dollars.

D. Multi-Process & Hybrid Testbench

In this case study, we describe the use of DMTCP's support for checkpointing distributed processes for two different multi-process simulation test benches.

In the first scenario, we discuss multi-socket simulation use cases. Here the simulation of each CPU socket is delegated to a separate simulator process. The simulator processes can be distributed across different computer nodes for reasons of efficiency. The communication between these simulator processes can be modelled on top of different inter-process communication mechanisms such as network sockets or shared-memory areas.

The second scenario where distributed checkpointing is used is that of hybrid test benches. Here the simulation model is broken into multiple parts and each portion is modelled differently. For example, a processor core can be modelled using a C++ program and some IP/PCH can be modelled as an RTL model, with the two portions communicating over a TCP socket connection.

IV. Results

There are two main achievements that are an outcome of this work.

- Improving the efficiency of different types of simulation regression.
- Leveraging the close collaboration with academia for solving the critical limitations in existing tools.

Table II describes the savings in compute time and, in some cases, reduction in turnaround time. This is achieved by skipping the full reset flow, or a part of it, using event-based checkpointing. The first run that creates the checkpoint takes a little longer to complete because of the checkpoint creation overhead. However, subsequent runs

Agnisys Technology.

that use the restore flow take much less time to complete. This is extremely useful in cases where multiple tests have a standard model configuration, test options, and initialization sequence. The main gains are achieved when the test content can be changed without the need to rebuild the model and without the need to run with different pre-conditions and configurations. In Table II, SoC1 and SoC2 are the best examples for this scenario. For both of these cases, the test content is read from a file that is not built as part of the simulation model.

Table I. Savings achieved with DMTCP

SoC	Without DMTCP	With DMTCP first run	With DMTCP subsequent runs	Save Restore Overhead	Disk space / checkpoint	Savings
SoC 1	48 hours	48.5 hours	2 hours	25 mins	10 GB	45.5 hours
SoC 2	54 hours	54.5 hours	12 hours	25 mins	10 GB	41.5 hours
SoC 3	12 hours	12.5 hours	2 hours	30 mins	3 GB	9.5 hours
SoC 4	15 hours	15.5 hours	4 hours	30 mins	10 GB	10.5 hours

SUMMARY

This paper shows that checkpoint-restore using DMTCP has definitely added value to the overall simulation flow. It successfully achieves its objectives in the use cases where the model configuration and code does not change between save and restore. Our results show that there can be up to 95% savings in the simulation time, with an average savings of 50% to 60% in the simulation time. Without checkpoint-restore using DMTCP, a traditional approach would have consumed decades of computation time, and this would have limited the number of possible tests. Therefore, the turnaround times can be greatly reduced in order to achieve 100% content validation. For other use cases, such as volume validation, skipping of initialization sequence is not possible. This is because each test has its own unique and fixed configuration that cannot be changed during restore. In such scenarios, checkpoint-restore can still be used to improve the reliability of the simulation runs and speed up debugging.

ACKNOWLEDGMENT

We would like to thank the full DMTCP team at Northeastern University for the close collaboration and partnership in this project. We would also like to thank all the users who helped, starting from initial conception till the development of the components of this new tool. Last but not least, we would like to thank Agnisys technology for sponsoring and would like to thank all the people who reviewed the paper and provided valuable feedback.

REFERENCES

[1] J. Ansel, K. Arya and G. Cooperman, "DMTCP: Transparent checkpointing for cluster computations and the desktop," in IEEE International Symposium on Parallel & Distributed Processing, Rome, 2009.

[2] J. C. Duell, E. Roman and P. H. Hargrove, "Berkeley Lab Checkpoint/Restart for Linux," Lawrence Berkeley National Laboratory, 2003.

[3] Ljubuncic, Igor; Rozenfeld, Avikam; Goldis, Andrew and Giri, Ravi, "Be kind, Rewind – Checkpoint & Restore capability for improving reliability of large scale semiconductor design" IEEE, 2014.