cadence®

Compressing Datasets Created During Silicon Design

By Guru Rao, Distinguished Engineer; Shakir Abbas, Software Engineering Group Director;
Mohammad Mirfendereski, Configuration Management Architect; Cadence
Harsh Sharangpani, CEO and CTO; Rajesh Patil, VP-Business Development; Ascava

During the design cycle for modern semiconductor components, a very large amount of data is generated and stored, often accumulating to hundreds of terabytes. Traditional compressor solutions are inadequate, and data footprints remain unwieldy and expensive to manage. To address this issue, Cadence evaluated a compressor called the Ascava App. This app addressed the problem of reducing large data footprints and can be incorporated into the infrastructure flows of enterprises to significantly cut data storage and communication costs and to improve transfer speeds on datasets created during the silicon design process.

Contents

Introduction

The design cycle for contemporary semiconductor components typically ranges from a few quarters of a year to a couple of years. During this period, the design moves through various phases, including microarchitecture definition, logic design, circuit design, and layout design, while simultaneously optimizing across functionality, timing, power, and area to converge the final design. During each phase, progressively refined models of the design are repeatedly created and iterated upon. A large amount of data is generated at each step, much of which needs to be retained for the duration of the project, and often for much longer.

Designs usually incorporate building blocks and libraries provided by vendors and foundries. These libraries themselves go through an extensive design and characterization phase, creating a large data footprint for their representation and characterization. Due to these factors, the storage requirements for semiconductor design cycles often exceed hundreds of terabytes. Vendors and foundries that distribute libraries to hundreds of customers face the challenges of moving many petabytes of data annually. At the same time, costs to store data (either on-premises or in the cloud) continue to be high, and distribution over WAN optimization solutions or content delivery networks is prohibitive. It becomes imperative to employ an efficient data compression technique to reduce costs associated with data storage and communication and to increase manageability and speed of transfer.

The Ascava App and Data Distillation Technology

Recognizing the data footprint challenges faced by design houses, Cadence has evaluated a compressor from Ascava, called the Ascava App.

The app exploits redundancy along new axes beyond existing compressors. While the Huffman method re-encodes literals based upon their dynamic frequency of occurrence (entropy) in a message, and the Lempel-Ziv method replaces copies of strings with a pointer to the original occurrence of the

string, Ascava's Data Distillation® Technology replaces portions of data with programs that derive the data from building blocks called "prime data". The gzip compressor implements a combination of the Lempel-Ziv and Huffman re-encoding methods and is well-recognized and widely deployed.

The Ascava App implements a combination of the Data Distillation Technology and the traditional methods of Lempel-Ziv and Huffman re-encoding to losslessly deliver the smallest data footprint. The Data Distillation Technology exploits redundancy across a scope that is several orders of magnitude larger than the window of traditional compressors. The Ascava App is designed to ingest large datasets comprised of multiple files and directories and is capable of exploiting redundancy across the scope of the entire dataset. The Ascava App provides 11 levels of settings for reduction; a higher level is designed to deliver higher reduction at the expense of throughput.

Cadence evaluated the Ascava compressor on three categories of data:

- Library characterization data generated by the Cadence® Liberate™ characterization engine
- A GDSII file (note that GDSII is the industry-standard format for interchange of integrated circuits and their layout artwork)
- Perforce checkpoint files from the Cadence Configuration Management group that maintains and frequently backs up the Cadence tool code base

The experiments were conducted, and the results of the findings are described below.

Experiments on Library Data Generated by the Liberate Characterization Engine

The Liberate characterization engine is a high-performance and ultra-fast engine used for creation of standard cell and IO libraries. It generates electrical cell views for timing, power, and signal integrity. With increasing number of corners that are characterized at advanced nodes, the size of a library database has increased exponentially. Cadence and Ascava conducted two experiments on library characterization data generated by the Liberate characterization engine.

First experiment

Library characterization data was generated for 475 cells across 11 PVTs (corners of process, voltage, temperature) with four library files (.lib) created per PVT. The 4 .lib files created contained CCS-timing, CCS-noise, ECSM and NLDM views for timing, power and signal integrity analysis. The resulting footprint of the four .libs for a single PVT was 3,984,588,800 bytes. For designs that need numerous PVTs (e.g., 100 corners), the overall footprint of the characterization data would increase proportionately from 4GB to approximately 400GB for this library. Performing this characterization four times a year would create 1.6TB of data for this library alone. Note that although only four views were created for this experiment, the Liberate characterization engine is often used in design flows to create more than four views, resulting in even larger data footprints.

The output of the characterization for a single PVT (the four .lib files) was resident in a directory that was fed to the Ascava compressor and the size of the compressed data was recorded. The Ascava compressor can ingest a directory and exploit redundancy across all files under the directory. Separately, a tar file of the directory was created and filtered using the gzip compressor, and the size of the resulting compressed tar file was recorded. The size of the compressed data generated by each of Ascava and tar+gzip and the time taken for compression were tabulated. This was repeated across all PVTs. The run for the Ascava compressor and the run for tar+gzip were each constrained to a single CPU core.

Table 1 summarizes the results of experiments comparing the compression performance of gzip versus the Ascava App on this data. Upon ingesting this data, gzip v1.5 at level three reduced the data to 666,894,336 bytes, yielding a reduction ratio of 5.97X. The Ascava App run at level three (in in-memory mode) reduced the data to 320,864,256 bytes, yielding a reduction ratio of 12.42X, which is more than 2X the reduction achieved by gzip. Ascava App level three is 1.23X faster than gzip level three at compressing the data.

Experiments conducted at: Cadence, by Cadence Liberate team in January 2019
Machine used: RHEL 6.5, Intel® Xeon® CPU E5-2699 v3 @ 2.3 GHz (Haswell generation)
Dataset: LIBARCH (11 PVTs, 475 cells, 4 .libs/PVT); Original size: 3,984,588,800 for 1 PVT

Tool	Level	Memory Quota	Compressed Size (bytes)	Compression Ratio	Compression gain Ascava over gzip	CPU Seconds (user+sys)	Compression speed (MB/sec)	Speedup Ratio over Gzip
Gzip v1.5	3		666,894,336	5.97		70	56.92	n/a
Ascava	3	5G	320,864,256	12.42	**2.08X**	57	69.91	**1.23X**

Table 1: Compression performance of Ascava App versus gzip on the Liberate Characterization dataset #1

The results in Table 1 were for characterization data for a single PVT. For multiple PVTs, the overall input footprint is proportionately larger. Note that the Ascava App has the potential of exploiting redundancy across the various .libs in this expanded footprint, while gzip is unable to exploit redundancy beyond the scope of a limited window (typically 32KB); such a limited scope of gzip is unlikely to span multiple files.

Second experiment

Library characterization data was generated similarly to the first experiment described above and the resulting footprint of the 4 .libs for a single PVT was 1,793,097,099 bytes. The total footprint including all directories and files for 100 PVTs that were generated is 181,879,618,099 bytes.

The methodology followed for this experiment was like that described in the first experiment above. Table 2 summarizes the results of experiments comparing the compression performance of gzip versus the Ascava App on this data. Upon ingesting this data, gzip v1.5 at level six reduced the data to 415,434,368 bytes, yielding a reduction ratio of 4.32X. The Ascava App run at level three (in in-memory mode with a memory quota of 1.5GiB) reduced the data to 201,570,028 bytes, yielding a reduction ratio of 8.9X, which is 2.06X the reduction achieved by gzip. Also, note that the Ascava App level three ingested the data at a compression speed of 107.50 MB/s, which is 3.62X faster than gzip level six.

Experiments conducted at: Ascava. Data by Cadence Liberate team in December 2018
Machine used: RHEL 7.3, Intel® Xeon® CPU E5-1650 v3 @ 3.5 GHz (Haswell generation)
Dataset: LIBARCH (100 PVTs, 4 .libs/PVT); Original size: 1,793,097,907 bytes for 1 PVT

Tool	Level	Memory Quota	Compressed Size (bytes)	Compression Ratio	Compression gain Ascava over gzip	CPU Seconds (real)	Compression speed (MB/sec)	Speedup over Gzip
Gzip v1.5	6		415,434,368	4.32		60.39	29.69	n/a
Ascava	3	1.5G	201,570,028	8.90	**2.06X**	16.68	107.50	**3.62X**
Ascava	4	1.5G	193,602,543	9.26	2.15X	35.25	50.87	1.71X

Table 2: Compression performance of Ascava App versus gzip on the Liberate Characterization dataset #2

Experiments on GDSII Data

A file containing the GDSII representation of a design was compressed by the Ascava App and by gzip.

Table 3 summarizes the results of experiments comparing the compression performance of gzip versus the Ascava App on this GDSII data. Upon ingesting this data, gzip v1.5 at level six reduced the data to 35GiB, yielding a reduction ratio of 4.80X. The Ascava App run at level four (in in-memory mode) reduced the data to 6.77GiB, yielding a reduction ratio of 24.81X, which is more than 5X the reduction achieved by gzip. Also, note that the Ascava App level four was 4.29X faster than gzip level six at compressing the data and 3.57X faster than gzip at decompressing the data. The retrieval rate achieved by the Ascava App was 428.57 MB/sec on this data.

Experiments conducted at: Cadence, by Cadence Open Access Database Group, July 2018
Machine used: RHEL 6.5, Intel® Xeon® CPU E5-2697 v4 @ 2.30 GHz (Broadwell generation) 18 cores
Dataset: GDSII file, original size: 168GiB

App	Level	Memory Quota (GiB)	Reduced Size (GiB)	Reduction Ratio	Ingest Time (h:m:s)	Ingest Speed (MiB/s)	Retrieve Time (h:m:s)	Retrieve Speed (MiB/s)	ascava reduction gain over gzip	ascava ingest speedup over gzip	ascava retrieve speedup over gzip
Gzip v1.5	6	n/a	35	4.80	3:32:40	13.17	0:23:21	119.91	n/a	n/a	n/a
Ascava	4	48/48	6.77	24.81	0:49:35	56.47	0:06:32	428.57	**5.17X**	**4.29X**	**3.57X**
Ascava	3	48/8	17	9.88	0:28:24	98.59	0:16:06	173.91	2.06X	7.49X	1.45X

Table 3: Compression performance of Ascava App versus gzip on GDSII dataset

Experiments on Cadence Configuration Management Data

A Perforce checkpoint data file was compressed by the Ascava App and by pbzip2 (Parallel bzip2). In both cases, four parallel threads were used to compress the 417GiB input dataset.

Table 4 summarizes the results of experiments comparing the compression performance of pbzip2 versus the Ascava App on this data. Upon ingesting this data, pbzip2 at level nine reduced the data to 21GiB, yielding a reduction ratio of 19.86X. The Ascava App run at level seven (in in-memory mode) reduced the data to 11GiB, yielding a reduction ratio of 37.91X, which is almost 2X the reduction achieved by pbzip2. Also, note that the Ascava App level seven running on four concurrent threads compressed the data 2.34X faster than pbzip2 level nine running on four concurrent threads.

Experiments conducted at: Cadence, by Cadence Configuration Management team in April 2018
Machine used: RHEL 6.5, Intel® Xeon® CPU E5-2680 @ 2.7 GHz (AVX, IvyBridge generation)
Dataset: Perforce checkpoint file, original size 417 GiB

App	Mode	Level	Memory Quota (GiB)	# threads	Reduced Size (GiB)	Reduction Ratio	Ingest Time (hr:min)	Ingest Speed (MB/s)	ascava reduction gain over pbzip2	ascava ingest speedup over pbzip2
pbzip2		9		4	21	19.86	4:27	27.94	n/a	n/a
Ascava	Inmem	7	64	4	11	37.91	1:54	65.44	**1.91X**	**2.34X**

Table 4: Compression performance of Ascava App versus pbzip2 on CM Perforce checkpoint data

Conclusion

Cadence studied the performance of the Ascava App on several datasets including cell library characterization data generated by the Liberate characterization engine, a GDSII file, and Perforce data from the Cadence configuration management backup infrastructure. On these datasets, the Ascava App delivered reduction levels of 12X, 25X, and 38X respectively, which is between 2X to 5X the reduction delivered by existing compressors (gzip, pbzip2). The Ascava App was consistently faster than the existing compressors. In silicon design flows that use the Liberate characterization engine, the Ascava App was found to be as easy to incorporate as using existing compressors.

In Cadence sample runtime scripts to control the Liberate characterization flow, Cadence has included the use of the Ascava App to reduce data produced by the flow. Based on its overall benefits and ease of use, Cadence has bundled the Ascava App with Liberate version 19.2 and has made it available to its customers for trial use.