SiliconIP

We Compress and
Process Your Data

Dr. Dan Tamir, CEO / CTO

- Associate Professor of CS at Florida Tech. & TX State Univ.
- Engineering and leadership work at Motorola, StarCore, Freescale, and Tadiran
- 33 years of data compression research experience
- DSP Architecture manager at Motorola/StarCore/Freescale
- Member of the MPEG committee and the IMS-Elite Expert Service

SiliconIP

- SiliconIP has developed a portfolio of novel algorithms along with architecture level design and RTL for compression engines, decompression engines, and compressed domain processors.
  - The algorithms allow for better compression than known algorithms, along with a lower implementation cost, lower latency, lower energy consumption and higher throughput
  - Some algorithms maintain short history buffers, allowing for very low-latency, pipelined implementations. Additionally, these algorithms provide compression that is better than competitive algorithms.
  - Other algorithms maintain longer history buffers (dictionary). The history buffers used by SiliconIP are significantly smaller than those used in LZ-based compression. Hence, they induce architectures with lower complexity (area) and shorter latency than classic implementations
- SiliconIP has also developed the concept of the *Compressed Domain Processor*
  - For certain algorithms, a CDP allows computation with streams of compressed data without the need to decompress, compute and recompress

Business Model

- SiliconIPs' business model centers around strategic partnerships with OEMs, and suppliers converging around its IP and next-generation product mix of data compression, processing, storage, and transmission technology.
- By licensing an array of "off-the-Shelf", highly specialized, customizable, SiliconIP solutions to design-, assembly-, manufacturing-, and implementation- partners, we accentuate our partners ability to deliver faster, more robust, cost effective, go-to-market solutions to their direct customer base.
- With continuous R&D since 2014 and an expanded patent portfolio in 2019, we deliver unparalleled value with highly attuned focus in the following disciplines:
  - Unsurpassed search, retrieval, & storage operations with voluminous disparate data
  - Industry leading processing, acceleration, and transmission of vast data streams
  - Unrivaled IoT sensor and AI data performance solutions with hyper-scale data loads
  - **Green** data center technology, significantly reduced cooling and compute costs
  - Strategy, scoping, design, integration, implementation, consulting, and post-production support

Presentation Goal

- The goal of this presentation is to further explore the technical aspects, industry standards, and benchmarks associated with SiliconIPs' next generation of compression technology solutions and product mix.
- By better understanding the science behind present day restrictions and their respective "acceptable" standards, one can effectively develop solutions that precisely target the challenges and limitations of today's and tomorrow's technologies.
- The enclosed information would enable stake holder to obtain enhanced data solutions, through collaboration and leading scientific knowledge and research.

The Problem

- There are many known compression algorithms.
- Unfortunately, no one size fits all
  - they differ in achievable compression ratio
  - they compress different data types by different amounts
  - With better compression ratios, they tend to have longer latencies
  - better compression tends to require more hardware
- And often it is desirable to process compressed data
  - decompress, process, compress can be long and energy intensive

SiliconIP's Solutions

- SiliconIP offers several new compression components that provide better performance than industry standards.
- Our portfolio Includes:
  - Compression Engines (CE)
  - Decompression Engines (DE)
  - Compressed Domain Processors (CDP)
  - Additional Intellectual Property
    - Innovative Random Number Generators
    - Pack/Unpack engines
    - Unbounded-integers compression
    - A "move to front" unit that combines least recently and least frequently used considerations in the reordering process.

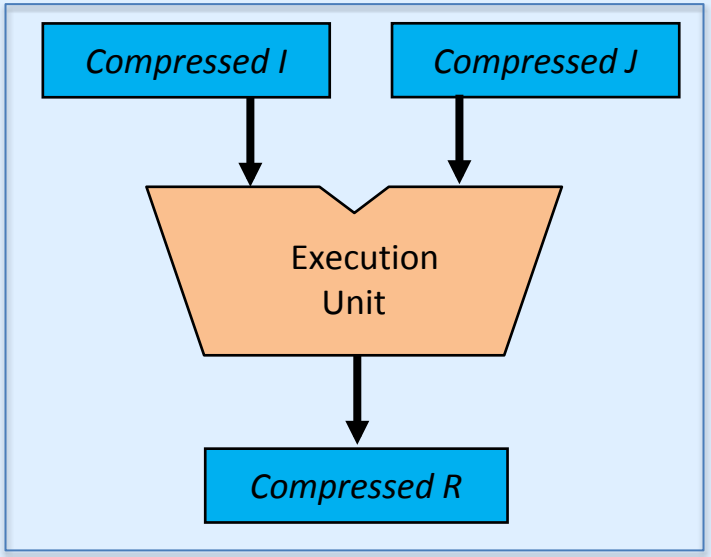
CDP – Compressed Domain Processor

- The standard approach to processing compressed data is to
  - decompress,
  - process,
  - recompress.

Our Approach: **process the data in its compressed form**

- The CDP can operate on compressed integer sequences such as sensor readings and posting lists. It can be:
  - as simple as providing ALU operations
  - as advanced as a general processing unit
- The CDP enables:

✓ arithmetic logical operations,
✓ searching, ✓ matching, ✓ sorting,
✓ encryption support,
✓ statistical analysis



A Rich Field: Market Players

- Information Retrieval
  - Internet    
  - Database  
- Communication and High Performance / Parallel Computing
  - Network Communication     
  - HPC / Intra & inter processor communication & storage   
  - Communication companies     
- Multi-media, Gaming, Virtual Reality, and Mobile Systems
  - Lossless compression components of standards (e.g., H.264)   
- Military, Defense, and Medical   
  - Encryption of compressed data
  - Support of factoring algorithms 
  - Genome data compression 

Data Types

- Our algorithms are used for compressing and decompressing two sorts of data
  - Integer sequences
    - this means data which is a sequence of values in which successive values have some correlation. Data from sensors is one example; certain information retrieval sequences also have this property[1]
  - Symbols
    - this means ‘general data’, such as the contents of memory or the contents of general files. Basically, we know nothing of the properties of the data, and successive values are to all intents and purposes uncorrelated. We may need ‘dictionary’ algorithms to compress this class of data well

[1] *https://static.googleusercontent.com/media/research.google.com/en//people/jeff/WSDM09-keynote.pdf*

Main Algorithms

Portfolio

Algorithm	Description
SigBits	Integers are represented with a header denoting their size in bits
FarFetch	Variable to fixed length coding

Further Algorithm Information

Algorithm	SiliconIP Proprietary	Integers/ symbols	History Buffer Size	CDP Support	Design level
SigBits	Y	Integers	1-4	Y	C + A + SVHDL[1]
FarFetch	Y	Symbols	16-256	Y	C + A + VHDL

[1] ***C*** *denotes High level language-based simulation,* ***A*** *means Architecture-level design, SVHDL refers to synthesizable VHDL*

[2] *In all cases the algorithm implementation is a part of SiliconIP's portfolio*

Competitive Analysis: Integer Sequence

Parameter / CE	SiliconIP SigBits	Baseline GVI
Implementation	Hardware	Software
Symmetry	Encode/Decode	Decode only
Throughput	4 integers / cycle	2B integers / second
Latency	1 cycle	Millions of cycles
Compression Ratio[1]	2.5x	2.1x
Cost (gate count (GC))	< 25,000 gates	Cores/GPU/MMX
Power (Implied by GC)	Very low	Very high
Pack / Unpack	Supported	Not Supported
Random Access	Supported	Not Supported
Exchangeable CEs	Supported	Not Supported
CDP	Supported	Not Supported

- If it is known that the input consists of correlated integer sequences, we can consider algorithms such as SigBits.
  - SigBits is memoryless. Hence, they can be efficiently pipelined.
  - SigBits provides higher compression ratio than other integer compression methods, at significantly reduced level of hardware complexity.
- Otherwise, we can consider symbol compression algorithms such as FarFetch, ALDC, GZIP, ZLIB, etc. The performance of these algorithms is detailed in a previous slide.
- The above numbers do not assume pipelined implementation.

[1]Using inverted index files obtained from Wikipedia

Competitive Analysis: Symbol Data

Parameter / CE	SiliconIP FarFetch	IBM ALDC Accelerator[1]	IBM ZLIB Accelerator[2]
Implementation	Hardware	Hardware	Hardware
Symmetry	Encode/Decode	Encode/Decode	Encode/Decode
Throughput	1.5 bytes / cycle	< 0.2 bytes/cycle	< 0.06 bytes/cycle
Worst case Latency	2 cycles	100s cycles	1000s cycles
Compression Ratio[3]	1.5x	2.2x	3x
Cost (gate count (GC))	< 25,000	> 200,000	> 1,000,000
Power (Implied by GC)	small	Relatively large	large
Pack / Unpack	Supported	Not Supported	Not Supported
Random Access	Supported	Not Supported	Not Supported
Exchangeable CEs	Supported	Not Supported	Not Supported

- The LZ algorithm and its variants have lower throughput and higher latency than FarFetch.
  - The LZ algorithms throughput and latency figures appear in IBM's publications (c.f., footnotes 1 and 2). Furthermore, they seem to be cherry picked.
  - The numbers do not account for worst-case latency
    - It may be hundreds of cycles in ALDC and thousands of cycles in ZLIB.
  - These numbers are achieved by hardware that is several times more complex (expensive) than ours.
- The FarFetch performance figures correspond to a non-pipelined implementation.

[1] *https://domino.research.ibm.com/tchjr/journalindex.nsf/0b9bc46ed06cbac1852565e6006fe1a0/5c65533b5307db6785256bfa0067fc0c!OpenDocument*

[2] *https://www.ibm.com/support/knowledgecenter/en/POWER8/p8hcd/fcej12.htm* [3] *Based on the Calgary Corpus (http://corpus.canterbury.ac.nz/descriptions/#Calgary includes the Calgary Corpus)*

Integer Sequences

- Sequences of Integers with relatively high auto-correlation and/or relative high concentration of small integers
  - For example, the data comes from information retrieval (IR) applications[1], from AI applications, and from large sets of data sensors in an IoT environment.
  - The sequence might be held in memory, held in storage, or be passing though interconnect
- In our simulations, we use realistic integer sequences generated from IR and/or sensor data acquisition.
- The goal is to enable efficient analytics of the data via the reduction of communication bandwidth & storage space.
  - Analytics can be done with SiliconIPs' CDPs.

[1] *https://static.googleusercontent.com/media/research.google.com/en//people/jeff/WSDM09-keynote.pdf*

General data

- A set of symbols (bytes)
  - The data might be held in memory or storage, where increased compression/decompression latency is acceptable
  - General data on interconnect may be compressed by the short-history algorithms, providing lower latencies, but does not offer as good compression rations
- In our simulations, we use data from a standard benchmark suite, such as the Calgary Corpus.
- The compression goal is to reduce communication bandwidth and/or storage space.

Use Case 1: Information Retrieval (IR)

- In this use case, it is assumed that an internet crawler is periodically generating inverted index files. The files are stored in one or more shared file systems.
- If the amount of data is large the operations on the inverted indexes and the computation mig be distributed between numerous Computing Units (CUs); e.g., clusters.
- A user generates the query <+State +Texas> and a map-reduce set of operations is applied.
- This use case fits integer compression engines and CDPs such as SigBits

Process:

1. The crawler places inverted indexes (potentially modulated and compressed by a SiliconIP CE) on one or more shared file systems.
2. A set of CUs fetches the postings list of the term 'State' and of the term 'Texas' from the file system.
3. If the data stored is pre-processed e.g., via differential coding (referred to as modulation) an then compressed, it can be decompressed and demodulated via a SiliconIP DE.
4. A set of CUs performs the matching and ranking operations (intersection operation in the cas of a binary search).
5. The results of matching and ranking performed by different CUs are aggregated and provide to the user.

SiliconIP's CDPs can be used in lieu of steps 3 and 4.

Use Case 2: Sensor Data Analytics

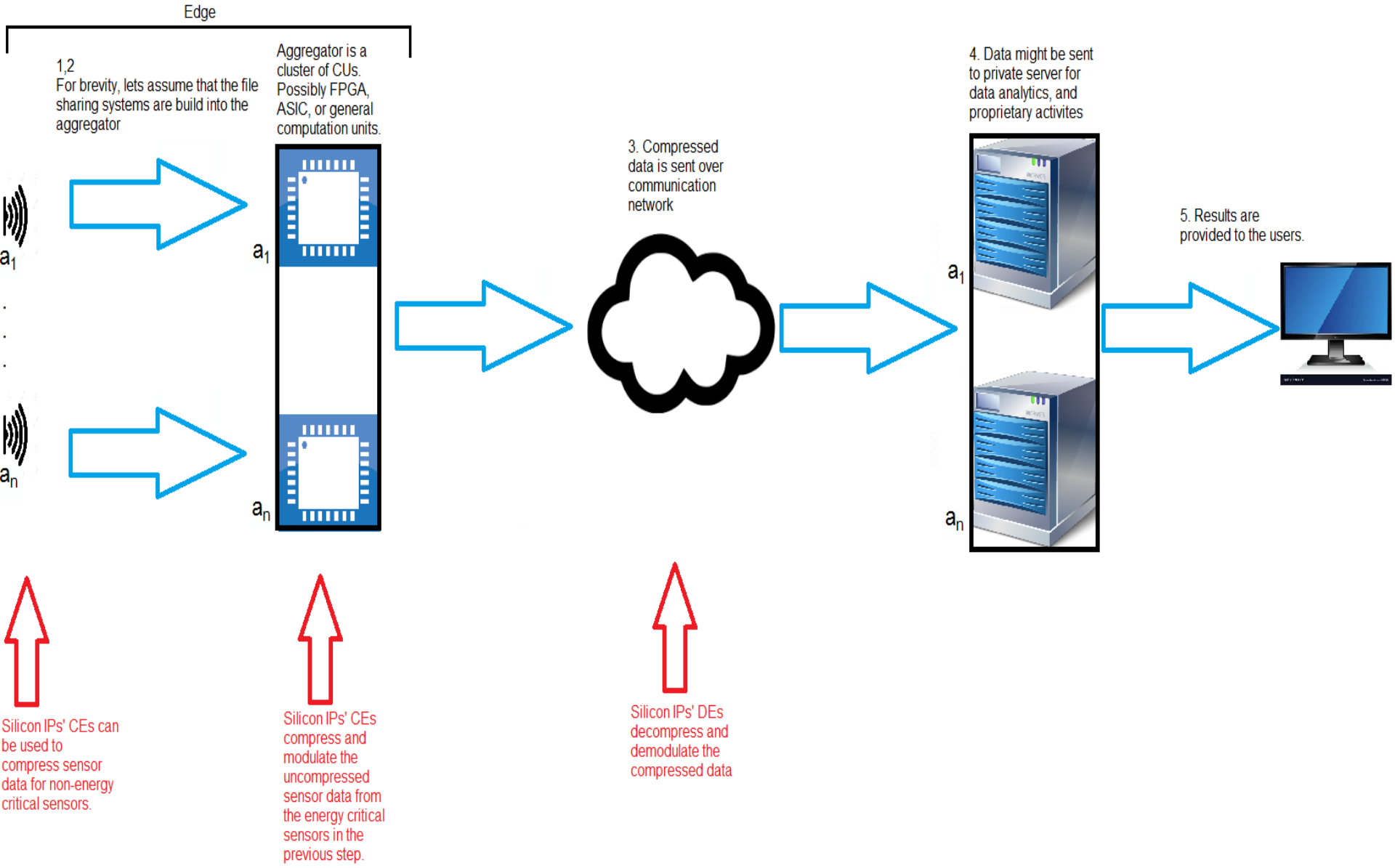
- In this use case, data is acquired by a large set of heterogeneous sensors and transmitted to a hub via a communication network (e.g., cellular communication network).
- These Sensors are divided into two types:
  1. Energy Critical – Those with energy efficiency requirements.
  2. Non-Energy Critical – Those with less stringent energy requirements.
- The none–energy critical sensors use a SiliconIP compression engine to compress the data prior to transmission.

Process:

1. Data produced by individual sensors is placed on shared file systems at regular intervals.
2. A set of CUs (e.g., clusters) is used to fetch individual sensor data and aggregate it. Here, a set of SiliconIPs' CEs compresses and modulates the uncompressed sensor data from the energy critical sensors in the previous step. At this point, all data is compressed and ready to be sent across the network.
3. The aggregated compressed data is distributed amongst a set of CUs (e.g. The Cloud). A set of SiliconIPs' DE units decompress and demodulate the compressed lists and the data is processed.
4. The processed results are further aggregated and distributed between a set of CUs for operations such as analytics, control, and anomaly detection.
5. Findings are provided to the users.

SiliconIPs' CDPs can be used in lieu of step 3.

Use Case 2: Sensor Data Analytics



Use Case 3: Reducing Bandwidth & Storage Space

- General data is transmitted through a communication channel and the goal is to reduce communication bandwidth.
- Alternatively, the data is stored in primary and/or secondary memory. The goal is to reduce storage space.
- There is no *a-priori* knowledge about the data.
    - It is handled on a byte by byte fashion where each byte is considered as a symbol.
- This use case can be handled with a FarFetch-based system. It is expected to provide moderate compression ratio along with high throughput and low latency.

Deployments

- IP Blocks implementing SiliconIP algorithms may be deployed in systems to provide compression/decompression on
  - Storage
    - For read-mostly storage (usually the case) if decode is much faster than encode, systems engineering is simplified since decode may be fast enough to avoid extensive buffering/caching. When data is known to be integer sequences, a short history algorithm may be used
  - Memory
    - Memory contains general data, and so a long-history algorithm is desired. Because Memory has both reads and writes, compression and decompression occur frequently with a severe latency penalty, requiring caching and memory management solutions
  - Interconnect
    - Because the data is general, a long-history algorithm is generally preferred. When data is known to be integer sequences, a short history algorithm may be used allowing compress/decompress IP blocks to be inserted into the data transport pipeline with no ill effects beyond a few clocks' delay

Summary

- SiliconIP holds disruptive intellectual property for lossless data compression
  - The IP is currently in the form of algorithms, architecture, RTL-based design, C++ simulations of the compression algorithms, and USPTO patents
- SiliconIP Accelerates:
  - Search, retrieval, storage, processing, analytics, and transmission in Information Retrieval Artificial Intelligence, Machine Learning , IoT, Storage, and Data Communication Systems
- SiliconIP significantly improves:
  - Compression ratio, throughput, and latency
- SiliconIP significantly reduces the cost of:
  - Energy consumption, design, implementation, and operations
- SiliconIP is used for:
  - Compression, decompression, analysis, processing, transmission, storage, and encryption of Integers and general data
- SiliconIP is used in:
  - Data Centers, Clouds, Edges, Communication Networks, and Data Sensor Devices

Additional Algorithm Portfolio

Algorithm	Description
SigBytes	Integers are represented with a header denoting their size in bytes
VLC-RNS	Variable Length Residue Number System
VLC MRS	Variable Length Mixed Radix Number System
Comma Coding	A unique bit combination used as a separator
Golomb & Rice Coding	Uniquely decodable encoding of results of division and modulus
In Between	Bit-replacement algorithm
1210	Bit-replacement algorithm
LDPC	Lossless Differential and Predictive coding ,
LVQ	Lossless vector quantization
UP&UP	Packing and unpacking results of compression/decompression
Collatz Tree Construction	Random Number Generator
FAR	Frequency and recency-based cache replacement algorithm
EntropyBits	Combinations of entropy encoding and UD integer encoding

Further Algorithm Information

Algorithm	SiliconIP Proprietary	Integers/ symbols	History Buffer Size	CDP Support	Design level
SigBytes	N^2	Integers	1-4	Y	C + A + SVHDL
VLC-RNS	Y	Integers	1-4	Y	C + A
VLC MRS	Y	Integers	1-4	Y	C + A
Comma Coding	N	Integers	1-4	Y	C + A
Golomb & Rice Coding	N	Integers	1-4	Y	C + A
In Between	Y	Integers	1-4	N	C + A
1210	Y	Integers	1-4	N	C + A
LDPC	N	Integers	1-4	N	C
LVQ	N	Integers	1-4	N	C

[1] ***C*** *denotes High level language-based simulation,* ***A*** *means Architecture-level design*

[2] *In all cases the algorithm implementation is a part of SiliconIP's portfolio*