The Big Data Revolution
Beautiful Servant or Dangerous Monster?

Anna M. Ravitzki, Vtool, Paris, France (anna@thevtool.com)

I. FROM SCIENCE ONTO DATA

The world is experiencing the revolution of information, humanity shifting the hegemony from Science onto Data. Just as the printing revolution once flooded the world with information, now cybernetic space is engulfing the entire planet with enormous amounts of information particles. We are living in an era where knowledgeability, facts, and big data have completely taken over, and control us all. Information technology has replaced the scientific method. We must doubt the necessity of this mass of information, and re-expose mankind to its core essence, to knowing. Our goal is to shed new light, using technological tools that enable mankind to regain the faculty of thought.

Data is built on postmodernist technology, and has become a cultural and epistemological value, while we use artificial intelligence to focus within the limits of our own knowledge. Data, as an idea, is a discourse within the realm of epistemology. When the idea of truth is connected to this discourse, the linguistic value can either be truth or false – the information is massive and often comes as a true value – for example, X (Google) knows that Y. The word knowledgeability guides the trueness of the information, and as any externalist will say, knowledge or data is related to relevant facts, which are obviously external to the soul of the being who holds this knowledge.

Data Screening Processor

At Vtool, we are highly aware of the human dilemmas that arise from coping with massive amounts of data and its screening. Yet, we do not want to view technology as the source of all evil, and we hold that every technological innovation raises technology up onto the highest level of philosophic contemplation. We flow between phenomenology and pragmatism that grasps the world as an ever-changing place, and we want to develop a tool that will adapt the scientific inventions and the masses of information to the current times, and to our exploration goals.

We are seeking a state in which our mind will not explode from the excess of information particles that it cannot contain. Instead, Cogita copes with this information. Cogita is a tool used as a supreme data screening processor, and at this current stage it is mainly applied for screening bugs.

Using Cogita, we can create a modification of all of this data. The 20th century philosopher Martin Heidegger reasoned the essentiality of tools for modern physics. Cogita is such a tool, and we are certain of its crucial necessity. This mediating tool creates a new optical vision, a vision that is mediated by the technology itself, much in the same way that the telescope once served Galileo in observing the moon.

Visualize the Data

Cogita is a technological pathway that controls the data, and creates measuring perceptions that do not forego or leave aside the human consciousness. By creating a post-phenomenologist revolution that sees in technology a basic necessity, Cogita creates a method that screens all the superfluous data away from the mind, to reposition at its core the original human faculty of thought.

Cogita is built of visual structures that help us create a perception of hermeneutics from the data structures. For us, this hermeneutics is a tool within Cogita, built completely of such visual structures that visualize the data. Cogita brings back the visual interpretation, placing the eyes at the center.
II. A Boy or a Girl - Screening the Data

Hippocrates, the great Greek physician is considered “the father of medicine.” He was the first to define a systematic method to monitor symptoms, conclude a diagnosis, and offer an appropriate treatment. It is astounding that no major advancement in medical examination has been made for over 2000 years since. Not until the 1958 paper by Ian Donald, John McVicar, and Tom Brown [1].

At that time, a doctor administering patients with abdominal pain could not do much, but put his hands on the patient and guess on what is happening and under the skin. The treatment was therefore based on many speculations, superstitions, and mostly - chance. Needless to say, this method of conjecture was rarely useful - and often enough quite harmful.

Ian Donald, a red haired Scotsman who had served in the Royal Air Force before becoming a gynecologist, decided to draw on his vast experience with radar technology from this army service. He felt a firm conviction that he can do much more for his patients than simply guessing what is happening inside the body based on their description of the symptoms. Teaming up with Tom Brown, a wonderful engineer, they modified some instruments used for detecting industrial flaws in ships, and were the first to apply sound wave technology for photographing a fetus inside a woman’s womb.

Their radical invention was used that very same year in Glasgow Hospital, with immediate positive clinical results. Today we take it as a standard matter of fact thing that we can just look into the unknown, non-invasively, safely, and with no side effects.

III. Cogita - The New Data Screening Revolution

Cogita is a data screening, processing, and visualization tool that targets ASIC and FPGA simulation-based debug. Cogita processes massive amounts of data from various sources (i.e. log files and waveform databases), and provides a clear visual root cause analysis. Cogita consists of several principles, as described below.
Unified Database

One of the major challenges of traditional batch debug is relating between two sources of the data, log file and waveforms. Debugging with a text editor for viewing the VCD file log and a waveform, forces users to decipher data that have different attributes:

1. Time progresses vertically in the log file, while for the waves it is represented horizontally.
2. The log contains text, yet the waveforms visualize strictly numerical values.

Cogita aggregates multiple data formats into a single coherent visual scenario. It visualizes horizontally, over a timeline, showing a combination of testbench and DUT events while blurring out the boundaries of different data formats. Engineers can easily keep track of the scenario without having to switch from log messages to waveform and vise-versa.

Data Screening and Processing

Debug is done in stages. Error messages that describe failures serve as starting points for debugging a failing test. The engineer then makes assumptions and validates them using the data from the log and waves, until the root cause of the failures is found. With the huge gate count in modern VLSI, verification engineers often fall off the mark, because they follow a misleading debug path due to wrong assumptions and conclusions. This is highly time-consuming, and produces unproductive debug cycles. As in any big data issue, the engineer raises an hypothesis and then queries the data to prove or disprove their assumption. The undeniable challenge is rooted in the extreme ratio that exists between meaningful data and redundant data.

We have found that in any given point of a debug process, there are no more than ten messages and waveform signals that are helpful for reaching the right conclusion. Cogita runs sophisticated data screening and processing methods that isolate these crucial messages and waveforms, as described in the examples that follow below (Section IV. Applications).

Visualization

Visualization is a key element in data analysis. As discussed at length in our previous paper [2], Cogita visualizes the screened and processed data in a way that enables everyone to intuitively understand.

In Cogita, the data source (log message or waveforms VCD) does not determine the visualization. The data is displayed in the best possible way and the most comprehensible alternative for every scenario. Both an RTL bus or a testbench integer value can be interchangeable, as values over a bus or by height. This is dramatic, and ensures an accurate, shorter, verification cycle.

IV. APPLICATIONS

Several practical examples and applications are presented below to illustrate the principles outlined above.

Example 1 - Waves and Testbench FIFO Visualization

The following example demonstrates how Cogita reveals the root cause of a testbench bug, using the data visualization of a FIFO underflow. Note that in this case, the error message itself did not indicate the FIFO underflow. Instead, a long debug process that would have been otherwise conducted to discover it. Cogita’s smart visualization scheme indicates the source of the problem instantly.

The following Cogita image represents a FIFO level modeling in the testbench over a passing test scenario. The Y-axis represents the FIFO level, the higher the bar the fuller the FIFO. The X-axis represents simulation time. This chart immediately clarifies that the FIFO is never completely empty, and once it reaches the threshold value (0xF in this case), it can be written. The thresholds are clearly visible.
FIFO Level - Passing Test

The blue chart below represents a failing scenario, in which the FIFO does not get filled at the threshold and therefore underflows (manifested in a wrap around), marked by the red error bar. A visual comparison between the two instantly leads the engineer to pinpoint the real problem. The abnormality pops into your eyes, thanks to the visualization. That same information exist in the log file but it would be extremely hard to track down by just reading the log file the traditional way.

FIFO Level - Failing Test

In the image below, visualization of the data can be done in several forms, regardless of whether an RTL signal or testbench variable is being traced. The two views complement each other, creating one unified understanding of the behavior.

Testbench Integer Displayed as a BUS and Cogita Chart

Example 2 - Debug by Tracing Values throughout the Unified Database

The testbench configures the DUT’s DMA and then injects data into the FIFO. The DUT is expected to generate DMA transactions to the internal DRAM. The DUT generates DMA address set according to the configuration, but the addresses are not visible across the DUT’s interface. Thus, they cannot be traced only by looking at the waveforms.

The test fails on a FIFO overflow. Investigating the cause requires analysis of multiple waveform signals and log messages. Cogita enables a much shorter debug process, by sparing the engineer the risk of deviating into irrelevant events.

1. Read the log error message: “FIFO Overflow”.

   ![Cogita Highlights the Error Message and the Time Point in which it was Asserted](image)

2. Verify coherent report of the monitor by loading the actual RTL FIFO overflow indication bit.

   ![DUT Bit is Asserted in the Same Cycle as Reported in the Log File](image)

3. The engineers suspect a problem in the DMA-generated addresses. They decide to check the last data injected into the FIFO in RTL, to find any correlation with the generated DMA address.
4. The data (bit 15:0) is 0xD0E0. Cogita finds that the address related to this data in the log messages is 0x3FE1_E432. The message is displayed by Cogita with the exact time point in sync with the waves.

5. The engineer now understands the accurate root cause: the address ends with 0x2, while the DMA may only generate 64 bit aligned addresses.

The huge advantage that Cogita offers is its capacity to show all of the above in one single clear image, screening all of the irrelevant data aside.

Example 3 - Pattern recognition

A chip, like any machine, works in patterns of repeating sequences. A pattern can include communication packets that the testbench injects into the DUT, bus transactions, FIFO shaping, or any other scenario. A common failure is identified as a sequence which deviates from the rest of the pattern. Analysis of a valid sequence and comparison to a failing one is a basic debug technique, but it is often difficult to achieve.

By using tagging and suffix tree manipulations [3] on the log, Cogita can automatically detect repeating patterns related to the failing message. In some cases, the engineer may have to provide some basic information to assist in structuring the relevant good patterns, such as a starting point, length, and end point. In other cases, Cogita does this completely automatically.

As an example, the steps below represent a good pattern of DMA access:
1. Write to DMA configuration registers (base_address, number of blocks, writes per block).
2. Write DMA On bit.
3. Inject data to FIFO (DMA will generate addresses and access to memory).
4. Wait up to 1000 cycles.
5. Interrupt bit is set by DUT.

The test runs 100 such frames. In three of them a non-aligned base address is configured due to a testbench bug, so the DMA transaction is halted and the interrupt is not set. Yet, the expected interrupt did not happen, resulting in an error. The root cause is an incorrect base address configuration, but the verification engineer has to investigate the case step by step until its discovered. With Cogita, the pattern is detected automatically, exposing the fact that a non-aligned base address is exclusive to all failed transactions.

Cogita analyzes the log file and identifies a set of messages as a repeating transaction, which is then presented to the engineers for them to label it as a valid (“good”) pattern:
1. emitter = [AXI_WR]; message = DMA CFG config REG: Base_ADDR, Data: 0x????????
2. emitter = [AXI_WR]; message = CFG DMA config REG: WR_PER_BLK, Data: 0x????????
3. emitter = [AXI_WR]; message = CFG DMA config REG: NUM_BLKS, Data: 0x????????
4. emitter = [AXI_WR]; message = CFG DMA config REG: DMA_ON, Data: 0x1
5. emitter = [INT_MON]; message = DUT interrupt bit set = TRUE

The patterns are then displayed visually on a timeline, for effective and immediate understanding of the test. The engineers may quickly grasp how many sequences completed successfully, and if they are running in parallel, what their duration is, and so on.

In our specific example, Cogita finds 97 such patterns, and identifies another three bad patterns in which message #5 is missing.

A good pattern example (with specific base address configuration):

1. emitter = [AXI_WR]; message = DMA CFG config REG: Base_ADDR, Data: 0xABCD_1230
2. emitter = [AXI_WR]; message = CFG DMA config REG: WR_PER_BLK, Data: 0x10
3. emitter = [AXI_WR]; message = CFG DMA config REG: NUM_BLKS, Data: 0x3
4. emitter = [AXI_WR]; message = CFG DMA config REG: DMA_ON, Data: 0x1
5. emitter = [INT_MON]; message = DUT interrupt bit set = TRUE

A bad pattern example (with specific base address configuration):

1. emitter = [AXI_WR]; message = DMA CFG config REG: Base_ADDR, Data: 0xABCD_123F
2. emitter = [AXI_WR]; message = CFG DMA config REG: WR_PER_BLK, Data: 0x10
3. emitter = [AXI_WR]; message = CFG DMA config REG: NUM_BLKS, Data: 0x3
4. emitter = [AXI_WR]; message = CFG DMA config REG: DMA_ON, Data: 0x1
5. MISSING

Cogita reports to the user that in all 97 good patterns the base_addr[1:0] == 0, while in all bad patterns it is not. According to the specification, BASE_ADDR must be 64b-aligned, and this indicates a configuration (testbench) issue that must be resolved. A long debug process is avoided.

By defining a sequence threshold, Cogita can highlight sequences of high similarity using the defined pattern. The sequences are then displayed for straightforward comparison of the deviation from the expected pattern, and the failures are immediately identified.

By creating a dataset from the sequence of frames, clustering algorithms can be utilized (such as K-means) [3] to identify similarity between multiple failing frames, including as non-aligned address, extra clock cycles in between, or other hard to detect deviations.

Example 4 - Players and Waves Suggestions

Cogita applies industry standard machine learning algorithms. We see significant results using unsupervised and semi-supervised algorithms focused on building datasets from the log, based on the first error message as a starting point. Apriori, k-nearest neighbor, Gaussian naive Bayes, and Principal component analysis [4] are already used to successfully suggest relevant queries and signals, while highlighting the relevant time point.
A minimum success rate of 80% proves to greatly increase the query abilities that Cogia offers the engineer.

Taking the following error message:

```
# UVM_ERROR [write_host_axi_B_t] @ 3600.0ns: {hntme_scoreboard.sv::5466} => Unexpected write resp!
```

Cogita analyzes the error message and creates a dataset of all messages, including [write_host_axi_B_t] as an emitter, and “write resp” as part of the message syntax. This dataset will usually be between five to ten dimensions, based on the log syntax of the specific environment. This dataset is analyzed for patterns and anomalies. Cogita will then suggest a few of the following players, based on clustering algorithms for the dataset:

- All write transactions by the [write_host_axi_B_t] emitter.
- Cluster the write data and write address and suggest players with address or data deviation.
- Detect recurring patterns based on the emitter [write_host_axi_B_t] as a starting point, and suggest players displaying emitters that break the pattern.

**Example 5 - Mathematical Manipulations on the Data**

Each Cogita player is a graphical representation of data. The data of each player is an array of values, representing numerical data that changes over time. This is also true for each bus or signal derived from the VCD file itself.

In the field of statistics and linear algebra, many common manipulations are known to screen data for highlighting the relevant information to the analyzer. Such manipulations can be a derivative for focusing on the changes only, or a log10 which focuses on the trend and less on absolute values. Cogita enables adding manipulation stencils for each player, giving an additional layer of comprehension.

In the example below, the user analyzes the behavior of DRAM address access. Each DMA frame has a base address, around 0x3FE0_0000, several blocks using a fixed gap, and a consecutive addresses in each block. The first image represents the player “as is”, i.e. the write address over time.

```
Write Address over Time
```

Due to the high offset of the data, it is almost impossible to grasp the address pattern in the test. The small drop right at the start represents a problem, but unless we zoom-in it may remain unnoticed, let alone understood.

The second image shows a derivative highlighting the data trends. Using this manipulation, the address drop is clearly indicated by the high peak. In addition, transitions between the blocks are traced by the smaller peaks that are noticeable across the pink line.
Derivative of the Write Address Next to the Actual Address

Example 6 - Dimensional Reduction

In many debug techniques, simply reducing one or two data dimensions may result in a clear representation that immediately highlights a failure. As a common example, the engineer may be interested in a series of values of RTL bus or a testbench variable, while screening the time element. Trying to compare two buses, in a standard waveform view, the data that flows into the DUT and the exact same data that flows out proves very difficult to discern, as is evident in the image below. Comparing the two signals or time-shifting them will not improve our understanding.

**AXI Write Data Comparison with Reduced Time Point Dimension**

Cogita presents the input vs. the output data as a list, reducing the time point dimension. This is how the differences are caught instantaneously.

V. **Benefits**

Cogita’s visualization platform gives verification engineers an enormous advantage in understanding complex test scenarios by being able to grasp huge log files.

This version of Cogita is only the platform on which we are building the structures that automatically screen big data processed with proven machine learning algorithms, to ultimately display the results visually in a revolutionary, straight-forward way. The engineer is finally freed from doing the tedious needlework of navigating step by step across all the different databases, because Cogita does all the heavy lifting for him.

VI. **Challenges**

Machine learning is a field which has its roots in the 1950s. It wasn’t until the last few years that major breakthroughs were made, catapulting the field into mainstream practices. The key developments that allowed this, are the massive boost in computer power and in data storage, which allow for huge datasets to be processed in record time.

Today almost every startup has access to unlimited processing power and AI infrastructures, hosted by Google and Amazon for a marginal price. Applying this to the field of verification engineering is proving hard, due to security standards and a relative slow adoption rate of new technologies.

To experience a real leap in the industry, we have to have the capacity to build huge datasets, preferably also through cross-tier partnerships between verification engineers, company executives, and project leaders. This may prove to be both a technical and a legal challenge, having to convince companies that they can rely on remote servers for hosting their encrypted data. But it will grant us the ability to build datasets of structured data that bring results with much stronger impact than described here, in this article. Imagine a bug detection algorithm that has the same success rate as Google’s spam detection algorithm!
VII. Future

Cogita is focused on a single test. We aim to give Cogita the power to read and process data from multiple tests, and/or a complete regression. These databases open up the vast possibility of analyzing and presenting the following features:

- Comparison between two or more tests.
- Pattern recognition in a complete regression, not only on a single test transactions or packets.

By adding the user and the project as columns on datasets used by machine learning algorithms, Cogita can learn the patterns of players over time, those that are more useful to one engineer over another, or even suggest players to one engineer based on the debug techniques of another person on the team. In any debug process, much effort is invested in social communication. Having proven techniques for dealing with the big data of one test is essential before making the jump to a full-scale project involving multiple team members.

VIII. Summary

Our guiding vision has always been to find a way for transforming big data to its core essence, to knowing. Cogita is already used by leading market players, and at Vtool we heavily rely on its robust debug abilities for our own internal verification projects. We are receiving highly positive reviews from our clients worldwide.

We are seeing astonishing results by working with different algorithms and charting. Among other visualizations, we are demonstrating the possibility of presenting the entire environment as an active force map, for grasping the architecture over time. Cogita is currently focused on the process of debug during chip verification, however we have already seen our clients apply Cogita for other diverse tasks, including network performance analysis, and freight management. We are currently experimenting its application in additional domains, including finance, medicine, and intelligence, with excellent results.

We aim to provide evidence of the Cogita as a perfect tool for screening, mapping, sorting, and visualizing data in a way that everyone can grasp instantly. As one of our clients recently phrased it best: “We found Cogita to be intuitive to use, an independent tool that does not force you to change your workflow, thus giving value within a short time.”

We would like to thank Dan Alexander and Hagai Arbel for their contribution to this paper.

IX. References