Finding the Last Bug in a CNN DMA Unit

Bruno Lavigueur, Dean Ke, Eric Rao, Fergus Casey – Synopsys
Achin Mittal, Pallavi Kumari, Michael Thompson, Roger Sabbagh – Oski Technology
Agenda

• Introduction to Synopsys CNN DMA
• Level-4 Formal Sign-Off Process
• DMA Case Study
• Results
• Conclusions
EV6x Vision Processor IP

- **Applications**
  - ADAS
  - Surveillance
  - AR / Gaming

- **Features:**
  - High performance
  - Low power
  - Small area

- **Components**
  - RISC CPU
  - Vector DSP
  - CNN Engine
CNN Engine

- Autonomous CNN engine
  - Performs OpenVX™ graph CNN functions

- Examples of functions enabled:
  - Object detection
  - Image classification
  - Semantic segmentation

- CNN DMA
  - Bulk data transfer between system memory and CNN local memory
  - E.g. fast loading of CNN layer weights
CNN Verification Methodology

- Simulation
  - UVM
  - S/W test cases
  - Module-level tests
  - CNN core-level tests
  - Coverage (code + functional)

- Emulation

- FPGA prototyping
Challenge of Finding the Last Bug

- **Parallelism**
  - Multiple cores
  - Multiple levels of data caches
  - Multiple levels of MMUs

- **Programmability**
  - Application specific use cases

- **Configurability**
  - H/W configurations
  - S/W configurations

![Graph showing verification time and effort with resistant bugs and long mean time between bugs](image-url)
Agenda

• Introduction to Synopsys CNN DMA
• Level-4 Formal Sign-Off Process
• DMA Case Study
• Results
• Conclusions
Formal Capability Maturity Model

Level 1
- Linting
- Auto Checks
- X-propagation
- Unreachability

Level 2
- Formal Apps

Level 3
- ABV Formal
- RTL Assertions
- Arbiter
- FIFO
- Handshake
- Bus Protocol

Level 4
- Block Sign-Off
- System Arch.
- Sign-Off
- System Deadlock
- Cache Coherence
- Network
- Throughput

Level 5
- Load/Store Unit
- Warp Sequencer
- Multi-Port Buffer Mgr
- Multi-Lane Aligner
- MAC Rx Block

Auto Formal

Auto Checks
Level-4 Formal Sign-Off Process

Quality of results depends on all 4 Cs

Constraints

Design Under Test (DUT)

Coverage (Code and Functional)

End-to-End Checkers

Complexity (Abstraction Models)
Agenda

- Introduction to Synopsys CNN DMA
- Level-4 Formal Sign-Off Process
- DMA Case Study
- Results
- Conclusions
**CNN DMA**

- **Components**
  - DMAC: DMA Controller
  - IDMA: Inbound to local memory
  - ODMA: Outbound from local memory

- **FV challenges**
  - Long transfer times
  - Multi-dimensional data volumes
  - Non-contiguous memory space
  - Compression mode
  - Performance requirements

- **Partition into 3 FV DUTs**
Data Integrity Checkers

- **Decomposition**
  - Two-part, data transport checker

- **Configurable checker**
  - Transfer mode defined in DMA descriptor

- **FIFO-style data checker**
  - Combined with data volume abstraction to deal with complexity
Address Correctness Checkers

• Decomposition
  – Two-part address checker

• Wide range of transfer modes leads to complexity
  – E.g. address calculation for 3-D and 4-D transfers

• Checkers predict the pointer update value and read/store address
  – Reference model of the checkers based on specification document
Compression / Decompression Checkers

- Decomposition
  - Isolate compression / decompression logic
  - Further decompose into two parts
    - Control state machines
    - Data transformation functions

- Added cut-points on the source and destination buffers on either side of the compression / decompression functions
  - Decreased the number of flops in the COI of checkers

- Checkers predict what the compressed / decompressed data is
  - In case of ODMA, checkers also predict the value of the associated metadata
Performance Checkers

For example, performance checker for IDMA is decomposed into following checks:

- **IDMA should request read data from System Memory when:**
  - System memory is ready to accept requests
  - Local Memory is not stalling store requests
  - More data is required to complete the transfer from System Memory

- **IDMA should send a store request to Local Memory when:**
  - Local Memory is ready to accept requests
  - Store data is available in the buffer

- **IDMA should signal that transfer is finished when:**
  - The last store is accepted by Local Memory

• Formally verify the performance requirements
  - Understand cycle-by-cycle behavior of IDMA
  - Decompose the high-level properties into smaller proofs
  - Manage complexity using existing abstraction models
Copy Volume Transfer Window

• Performance checks require tracking of the copy volume transfer window

• Example:
  - IDMA_transfer_window [#cycles] =
    \[
    n_{\text{constant\_cycles}} + \\
    \text{copy\_volume\_cycles} + \\
    \text{system\_memory\_stall\_cycles} + \\
    \text{local\_memory\_stall\_cycles\_when\_store\_data\_available}
    \]

Fixed overhead

Varies depending on transfer type; can be huge
Abstraction Models

- Default depth of analysis is insufficient
- Abstraction models provide short cuts to cover deep states
Copy Volume Abstraction

- Example: 3D data transfer with dimensions of 100x20x10 = 20k units
- For 3-cycles/unit transfer time, total time is 60k cycles

- Abstract copy volume counters for each dimension ($x, y$ and $z$) for both memories

- Constraints on counter values:
  - Legal range: Initialize to 0, cannot exceed max value given in DMA descriptor
  - Relationship:
    - $y$ counter can only toggle when $x$ is at max value
    - $z$ can only toggle when $x$ and $y$ are at max value
  - Counters for destination memory can never get ahead of the counters for the source
Orders of Magnitude Reduction

- The number of cycles required to complete the transfer is reduced
- Multiple paths to completion can be covered in just a few cycles

3-D transfer: x = 100, y = 20, z = 10
## Functional Coverage Improvement

<table>
<thead>
<tr>
<th>Example Scenarios</th>
<th>Cycles to cover \textit{without} Abstraction Models</th>
<th>Cycles to cover \textit{with} Abstraction Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>All IDMA counters updated twice</td>
<td>35</td>
<td>14</td>
</tr>
<tr>
<td>All ODMA counters updated twice</td>
<td>50</td>
<td>8</td>
</tr>
<tr>
<td>Register to track z-plane jump should be reset after being set</td>
<td>Not reached in 6hr runtime (&gt;63)</td>
<td>19</td>
</tr>
<tr>
<td>Compress data buffers are empty after getting full</td>
<td>Not reached in 6hr runtime (&gt;68)</td>
<td>7</td>
</tr>
<tr>
<td>Two compression metadata writes seen, one of which is due to buffers being full</td>
<td>Not reached in 6hr runtime (&gt;103)</td>
<td>20</td>
</tr>
</tbody>
</table>
Constraints Validation

• Constraints used as asserts in CNN core-level simulation
  – Find bugs in constraints, neighboring blocks or I/F spec

• Found a bug in DMA’s neighboring block
  – The ready signal on the write interface was not driven correctly

• Found a bug in the simulation reference model
  – DMA behavior for a certain transfer had changed but the reference model had not caught up to it yet
Agenda

- Introduction to Synopsys CNN DMA
- Level-4 Formal Sign-Off Process
- DMA Case Study
- Results
- Conclusions
Simulation-Resistant RTL Bugs Found

- 6 corner-case bugs found at a late stage

<table>
<thead>
<tr>
<th>Module</th>
<th># Bugs Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDMA</td>
<td>4</td>
</tr>
<tr>
<td>ODMA</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>6</td>
</tr>
</tbody>
</table>
Example Simulation-Resistant Bug

• Deadlock occurs when three interacting IDMA FSMs reach specific states at the same time

• Multiple preconditions required
  – System Memory is slow to respond to a read request from IDMA
  – IDMA has run out of decompression metadata
  – IDMA moves to the next step in the z-dimension for the data volume transfer

• If all 3 preconditions occur in the same cycle, then:
  – IDMA enters a state which renders it unable to accept new metadata
  – Leads to deadlock due to a circular dependency
Circular Dependency Deadlock

IDMA needs to decompress data in the buffer to complete an ongoing transfer to Local Memory.

Compressed data cannot be accepted from System Memory because the Buffer is full.

Data in the Buffer cannot be decompressed because there is no metadata available.

IDMA cannot accept new decompress metadata.

3 preconditions required to enter this state.
Specification Bugs Found

- Two implementation details which indicated an outdated/incorrect spec:
  - ODMA compression FSM was observed to make a state transition which was not documented
  - In certain modes, ODMA stored formatted data in the buffer, whereas the specification document specified that only unformatted data should be stored
Agenda

• Introduction to Synopsys CNN DMA
• Level-4 Formal Sign-Off Process
• DMA Case Study
• Results
• Conclusions
Observations of Formal Sign-Off

• DMA has very long bulk transfer times
  – Abstractions enable formal sign-off

• Formal controllability coverage confirms depth of analysis
  – Functional and code coverage

• Formal observability coverage confirms exhaustive set of checkers
  – Cone-of-influence and formal core coverage
Conclusions and Future Directions

• Formal sign-off of CNN DMA
  – Improved confidence in shipping CNN IP

• Uncovered bugs that simulation had not found after many millions of cycles of random stress testing

• Provides confidence that we’ve found the last bug
  – Exhaustive formal analysis

• Formal sign-off will be a growing part of the verification strategy
Questions?

THANK YOU