STE - the Primary Validation Vehicle for Processor Graphics FPU

M, Achutha Kiran Kumar V
Aarti Gupta; Rajnish Ghughal

CMI @ 9 Jan 2013
STE - the Primary Validation Vehicle for Processor Graphics FPU

M, Achutha Kiran Kumar V
Aarti Gupta Jr.; Rajnish Ghughal

CMI @ 9 Jan 2013
Purpose

• To demonstrate how STE validation methodology was effectively applied to validate a re-architected FPU in short runway GT project

• Demonstrate the effective utilization of formal methodology from the beginning of the project
Agenda

• Next Gen GT FPU Val risk
• Results
• STE Overview
• GT STE implementation Challenges
• Conclusion
Agenda

- Next Gen GT FPU Val risk
- Results
- STE Overview
- GT STE implementation Challenges
- Conclusion
## NextGenGT FPU Validation Challenges

<table>
<thead>
<tr>
<th>Activity</th>
<th>Challenge Posed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete re-architecture of FPU</td>
<td>Validate all uops within limited timeframe</td>
</tr>
<tr>
<td>RTL and C++ Checker concurrent development</td>
<td>Need an alternate validation methodology to check the coded RTL</td>
</tr>
<tr>
<td>New Requirement: IEEE compliance for precision and exceptions</td>
<td>Perfect methodology to check for precision and ieee compliance similar to CPU implementations</td>
</tr>
<tr>
<td>Increased scope of denormal handling for all precisions</td>
<td>Dataspase explodes by 2X</td>
</tr>
<tr>
<td>New FMA architecture</td>
<td>To verify Sea of multipliers implementation</td>
</tr>
<tr>
<td>Complex Programming capability</td>
<td>Need to verify all permutations with increased data space</td>
</tr>
</tbody>
</table>
Contemporary Methodologies at a glance

<table>
<thead>
<tr>
<th>Validation Technique</th>
<th>Methodology</th>
<th>Reference Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>DV#1</td>
<td>Dynamic validation of targeted interesting dataspace cases vectors generated</td>
<td>C++ based Ref model</td>
</tr>
<tr>
<td>DV#2</td>
<td>Dynamic validation of controlled random vector generation</td>
<td>C++ based Ref model</td>
</tr>
<tr>
<td>DV#3</td>
<td>Dynamic validation using standard random test bench features of System Verilog</td>
<td>C++ based Ref model</td>
</tr>
<tr>
<td>FV#2</td>
<td>Formal Verification using a standard industrial tool</td>
<td>C++ based specification</td>
</tr>
</tbody>
</table>

Need of the hour: A verification methodology that could meet the project timeline requirements

Solution: A Formal Verification Methodology suitable for proving Arithmetic circuits:

**Symbolic Trajectory Evaluation (STE)**
Agenda

• BDWGT FPU Val risk
• Results
• STE Overview
• GT STE implementation Challenges
• Conclusion
Operation “FV Bug Hunt”

What gave STE an edge over other verification methodologies in Next Gen GT?

- One Proof – many projects
- One Proof – Wider Coverage
- Proof ready before RTL and Fulsim
- Capability to mask unimplemented features
Bug Hunt Comparison

RTL bugs caught by methodologies

- STE, 169 (72%)
- DV1,44 (19%)
- DV2,11 (5%)
- DV3,6 (2%)
- FV1,4 (2%)
Division of 201STE found bugs

- RTL: 84%
- Refmodel: 12%
- Bspec: 4%

Bug Division

- Dataspce corner issues: 26
- IEEE flag issues: 31
- NaN handling: 42
- Specification issues: 12
- Functional problems: 68
- Instruction Dependency: 8
- Clockgating: 14
Agenda

• BDWGT FPU Val risk
• Results
• STE Overview
• GT STE implementation Challenges
• Conclusion
Symbolic Trajectory Evaluation (STE)

- A hybrid between a symbolic simulator and a symbolic model checker

- Used primarily for checking designs with large datapaths

- Combines 3-valued simulation (0, 1, X) with symbolic simulation (using variables instead of fixed values)
**STE INFRASTRUCTURE**

**Constraints**
on nodes n at different times t. Some input nodes will be driven symbolic value; some may be fixed (0/1); all others will be made don’t care (X)

**Specifications**
process Input node time information to get symbolic value for writeback ports at writeback time (src time + latency)

**Src_info (src_nodes, src_time)**

**Wb_spec (wb_nodes, wb_time)**

---

**Symbolic Simulator:**
Computes a symbolic representation for each node for each time (n, t)

**Symbolic checker:**
Checks if wb_ckt & wb_spec are equivalent

**STE**

**EXPL**

**Exploded/Hang – Complexity reduction**

**Refine Antecedent/ Xs in circuit/ Complexity reduction**

**BUG/Refine Consequent**

**IMPLEMENTATION PROVED!!**

---

14
CVE – The Repository

- CVE – Common Verification Environment
- Collation of all proofs
- Foster reuse of common proofs across projects
- Avoid “reinventing the wheel” again and again
- Project specific qualifiers for differential treatment
Agenda

• BDWGT FPU Val risk
• Results
• STE Overview
• GT STE implementation Challenges
• Conclusion
STE Deployment Challenge

CPU STE Infrastructure → [Image]

GT STE Infrastructure
An Exhaustive instruction format

- Gen graphics instruction set is compact but has a complex format

```
[pred] instr Cond mod (sat)
(execsize) dst {Accdst}
<srcmod> src0 {Accsrc}
<srcmod> src1
<srcmod> srcn
```

CPU Instruction Format

```
instr execsize dst src0...srcn
```
CPU infrastructure reuse challenges

GT’s Own flag handling
Source Modification for all sources involved
Saturation for Floats
Implicit/ Explicit Accumulator Source/Dest
CPU infrastructure reuse challenges

- Non uniform Denormal handling across precisions
- ALT Mode
- Different way of NaN Handling
- Instruction specific rounding modes
- HP and QW support
- New FMA Architecture / Implementation
Our Approach

• Added / Redefined common functions/fields in CVE
• Project specific qualifiers
• New proofs
• Complexity reduction techniques
• New Variable ordering
• New Data type support
• Infrastructure to support new implementations
Interesting bugs #1

(MAD-DP)

Multiplicand \( a \) = 0x1cc9_9398_0003_3273 = 1.xyz \times 2^{(-512)}
Multiplier \( b \) = 0x1ff4_04b2_5a15_c2bb = 1.abc \times 2^{(-563)}
Addend \( c \) = 0x8000_0000_0000_0001 = 1.0 \times 2^{(-1074)}

Product \( a \times b \) = 1.0 \times 2^{(-1075)}
Expected Result \((ab+c)\) = 0x0000_0000_0000_0001
Actual Result \((ab+c)\) = 0x0000_0000_0000_0000

Dataspace Corner case issue
Interesting bugs # 2

(MAD-DP)

Conditions on preceding instruction:
Operation must be MAD-DP and
Addend = Not INF/NAN/ZERO and
Addend is –ve

Conditions on current instruction:
Operation is MUL-DP
Multiplicand/Multiplier = -ve NAN

Expected Result =
ffff_ffff_ffff_ffff

Actual Result =
7fff_ffff_ffff_ffff

Instruction interaction bug
Future Plans

• STE on FPU for Future GT projects

• Apply STE on more datapath blocks..

• Improve the proof database to add more uops
Agenda

• BDWGT FPU Val risk
• Results
• STE Overview
• GT STE implementation Challenges
• Conclusion
Conclusion

• Next Gen GT FPU re-architected for optimizations, IEEE compliance and for improved programmability
• STE as the prime tool found 201+ bugs
• Validation prior to Ref model readiness and wider coverage.
• Lower Time/uop validation
• Reduction in overall Validation cost for datapath dominated designs
Acknowledgements

- Roope Kaivola – FVCOE
- Tom Schubert – CCDO FV Management
- Naveen Matam – uarch for EU
- Maiyuran Subramanian – Arch for EU
- Archana Vijaykumar, Durairaghavan Kasturirangan – EU/Val Management