RegAnalyzer -A tool for programming analysis and debug for verification and validation

Suresh Vasu
Engineering Manager and Senior Technical Lead
Suresh.Vasu@intel.com
Intel Technology India Pvt Limited, No 23/56-P, Outer Ring Road, Bellandur, Bengaluru – 560103

***Abstract*- As the technology node is shrinking, more and more IPs are getting integrated in single System-on-chip. In the System-on-chip, verification is getting more tedious, time consuming and challenging task. Based on case study on different internal and third-party IP's, realized that there are no mechanism nor internal / external tools available for doing the quick debug by understanding the test cases and its programming. Verification Engineers are putting manual effort to understand the IP programming in the test cases and there is no direct way to map the register programming to the Technical Reference Manual / Programmers model. This paper presents a tool RegAnalyzer which resolves this problem.**

I. INTRODUCTION

As the technology node is shrinking, more and more IPs are getting integrated in single system on chip. There are lot of industry standard or company in house developed RTL integration tools available for integrating the RTL. But in the System-on-chip, verification is becoming a more tedious, time consuming and challenging task. It involves a lot of third-party IP's and internal IP's for which different issues are encountered.

The presence of third-party IP in SOC has simplified the design while complicating the verification for the end user. The IP vendors generally don't share much design or architecture information with the end user hence making the job of verifying and understanding such IP's even more tedious and bound to dependent on the IP vendor to understand and resolve the issues. There are lots of manual efforts (from days to weeks) put in day-to-day verification activities which is consuming time and bandwidth of verification Engineers which directly affects the production cycle.

Based on case study on different internal and third-party IP's, we realized that there are no mechanism nor internal / external tools available for doing the quick debug and understanding of test cases. Verification Engineers are putting manual effort to understand the IP programming in the test cases and there is no direct way to map the register programming to the Technical Reference Manual or Programmers model. RegAnalyzer is the tool which is built to overcome the issues faced by the verification engineers and assist them for their debug.

II. VERIFICATION CHALLENGES

Figure 1 shows the percentage of total IC/ASIC project time spent in verification [1]. From the case study and analysis done by the team from Mentor, the complexity for doing the verification has increased significantly for past few years and the design size has also grown many folds. Due to many IP's are getting integrated in the SOC, verification engineers are facing lots of challenges in day-to-day debugging.

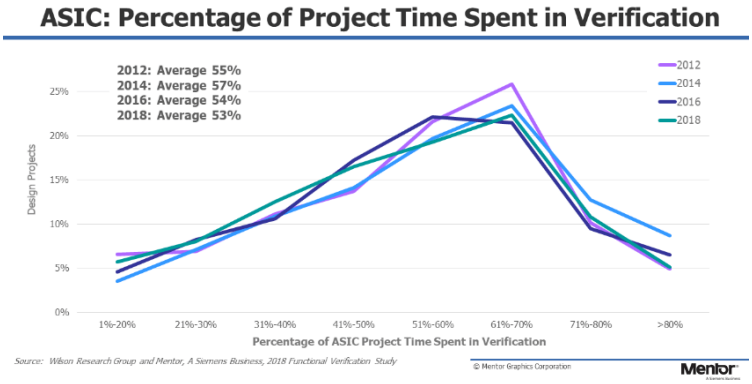


Figure 1. Percentage of IC/ASIC Project Time Spent in verification [1]

Figure 2 shows where verification engineers spend their time (on average). [1]. The study from the Mentor team shows that the verification engineer is spending more time in debugging during the chip development compared to all other verification tasks like test planning, testbench development, creating test cases, running simulation and support works. From the verification and validation engineer's experience, one of the aspects is for more debug time is due to not understanding the IPs register programming.

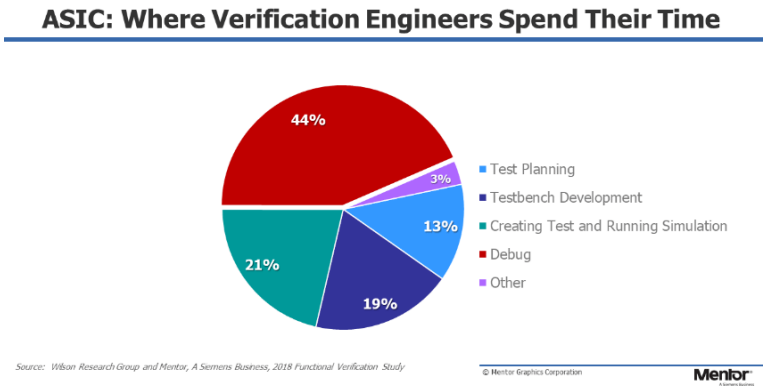


Figure 2. Where IC/ASIC Verification Engineers Spend their Time [1]

III. IP-XACT

IP-XACT is a standardization which is developed and maintained by the Accellera. IP-XACT is developed to capture the IP details in a standard specification and its written in a standard data exchange format (XML), which is human readable. Figure 3 describes about the format in which the register name, its offset and the register descriptions are defined in the IP-XACT file.

.

```
      <ipxact:register>
        <ipxact:name>STAT</ipxact:name>
        <ipxact:description>Status register. Collection of Status flags including interrupt status
before enabling</ipxact:description>
        <ipxact:addressOffset>'h0</ipxact:addressOffset>
        <ipxact:size>32</ipxact:size>
        ...
      </ipxact:register>
```

Figure 3. Component Register in IP-XACT [2]

Figure 4 shows the format in which the register field name, its description, value and its offset are defined in the IP-XACT XML file. For more details about the IP-XACT and its details, the user guide is added in the reference [2].

```
        <ipxact:field>
          <ipxact:name>RXFIFO_NE</ipxact:name>
          <ipxact:description>RX-FIFO Not Empty. This interrupt capable status flag indicates
the RX-FIFO status and associated interrupt status before the enable stage. The flag can only be
implicitly cleared by reading the RXFIFO_DAT register</ipxact:description>
          <ipxact:bitOffset>0</ipxact:bitOffset>
          <ipxact:resets>
            <ipxact:reset>
              <ipxact:value>'h0</ipxact:value>
              <ipxact:mask>'h1</ipxact:mask>
            </ipxact:reset>
          </ipxact:resets>
          <ipxact:bitWidth>1</ipxact:bitWidth>
          <ipxact:access>read-only</ipxact:access>
          ...
        </ipxact:field>
```

Figure 4. Component Register Field in IP-XACT

IV. REGANALYZER (3A - AUTOMATE, ANALYZE & ASSIST)

The RegAnalyzer tool uses the power of IP-XACT Component Memory Maps and Registers [2] and automation done on top of that. Figure 5 summarizes the problems in today's debug methodology. All the third-party IP's test cases, Validation collaterals, post silicon debug, issues reported by the software team and customers are getting resolved after decoding all the IP's programming manually. There is no straightforward way to understand the functionality verified or the intent of the testing scenarios. Figure 6 shows that RegAnalyzer can use test case logs or register dump or enter register values read through trace32 debugger, decodes it field by field and provides the report with its description also. Also, you can decode the programmed values different from IP's reset values to exactly know the feature enabled in the test cases / register dumps / values read through trace32.

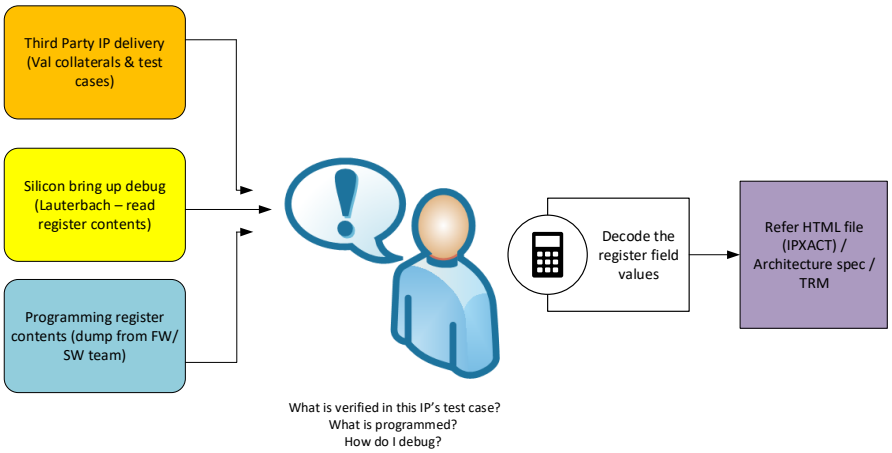


Figure 5. Manual efforts in Verification & Validation debug

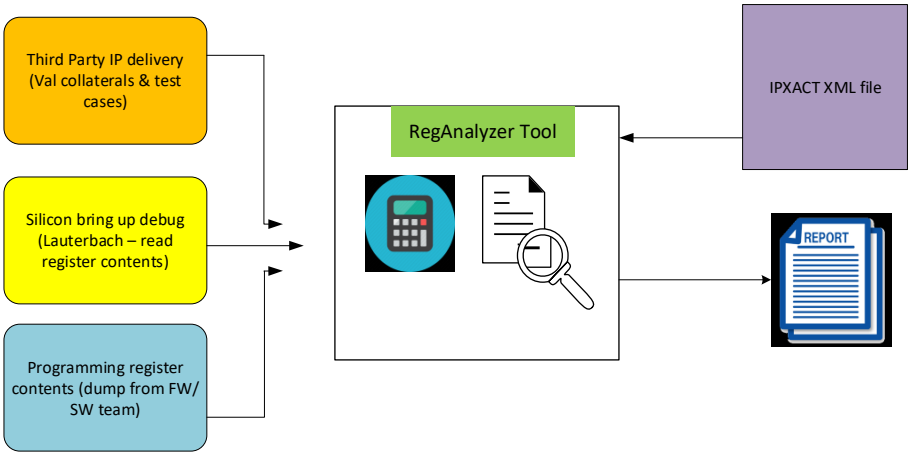


Figure 6. RegAnalyzer (Automate, Analyze & Assist)

A. RegAnalyzer Implementation Details

The RegAnalyzer tool requires three inputs - one of them is the IP-XACT XML file for an IP, log file / trace file / test case which has all register programming information and the address map. Figure 7 shows the RegAnalyzer GUI tool and its features. Figure 8 shows the one of the examples of the input which contains the register programming.

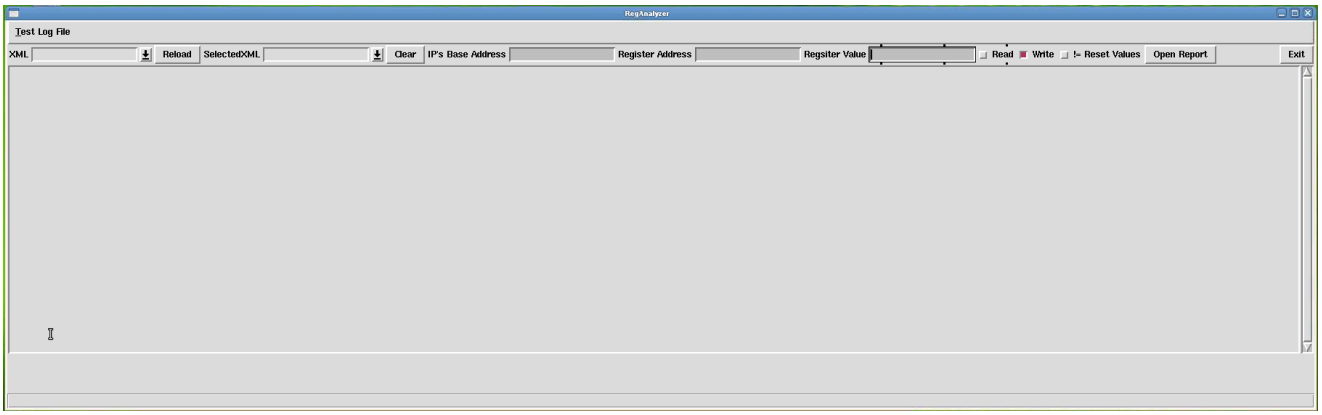
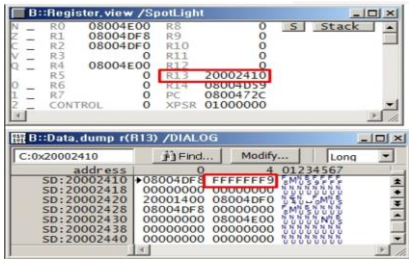


Figure 7. RegAnalyzer GUI tool

```
WRITE_WORD(0x00000004 , 0x00000000);
WRITE_WORD(0x00000008 , 0x00000400);
WRITE_WORD(0x00000008 , 0x00000400 );
WRITE_WORD(0x00000008 , 0x00000080);
WRITE_WORD(0x0000000c , 0x28004000);
```




```
W 0000000c/1800C000
W 00000010/01600900
W 00000014/10001A7C
W 00000018/0001A141
W 0000001c/0000003F
W 00000020/00000000
W 00000024/00000000
W 00000028/00000000
W 0000002c/00000000
W 00000030/00000000
W 00000034/00000000
W 00000038/00000000
W 0000003c/00000000
W 00000040/02031890
W 00000044/05050304
W 00000048/00000404
W 0000004c/10907D01
W 00000050/04030404
W 00000054/04040507
W 00000058/77020100
W 0000005c/11111510
W 00000060/00000001
W 00000064/11111130
W 00000068/00000111
```

Figure 8. Test case / T32 output / Trace file which contains the register programming

Once we load both the IP-XACT file and the log file which contains the register programming, the tool extracts all the register name, address offset, register fields and its description information from the IP-XACT file and dumps an intermediate file. Then, it unpacks all the 32 bits register programming values, packs the values depends on the register field size maps to the field offset and its fields description.

This tool provides the following features.

- Simple GUI interface
- Decodes the register programming from register dump (source may be from customers, silicon validation or software teams), display its register and field descriptions with programmed values
- Test Simulation log file / trace file analysis, mapping the register read or write to the individual register fields and their description.
- Option to decode only the programming field values which is different from the reset values and display it with description.
- Option to choose WO/RO/RW/ALL register's attributes.
- Option to choose the Read / Write access from the test log file.
- Option to open the report file
- Option to pass the IP's Base Address as per the SOC address map in case if the log file has a different base address.
- Option to enter the single register address and its programmed values. This will be useful in post silicon validation in which we read the register values using Lauterbach.

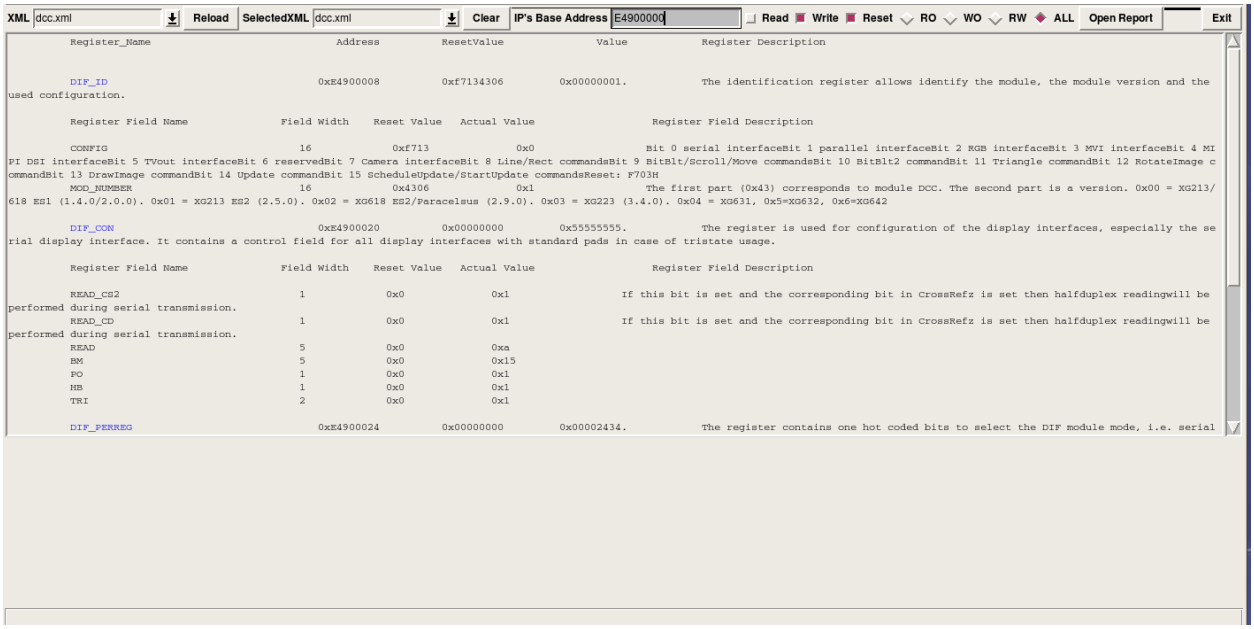


Figure 9. RegAnalyzer tool output

B. Results

The RegAnalyzer tool is implemented to decode the test cases during the latest chip development. There are lots of third party and internal IP's used in the SOC. For some of the complex Media IPs, we can run the IP delivered test cases, loaded the log files in the RegAnalyzer tool and generated the report on the programming details with the register fields description. Also, by using the no reset values feature, we generated the report of the programming register and fields which are different from the register reset values. By analyzing the report, we can understand the verification intent of the test case and features enabled in that test case. For some of the subsystem test cases, it

involves more that one IPs programming details. We provide the IPs address space details according to the SOC address map to the RegAnalyzer tool and all the IP-XACT XML files for those IPs. The tool analyzes the log file, finds out the register address from all the intermediate files generated from the IP-XACT XML files and finally generates the report. We have saved a lot of debug time while verifying these IPs in the SOC and it helped us immensely to understand the IP functionality by analyzing the programming details.

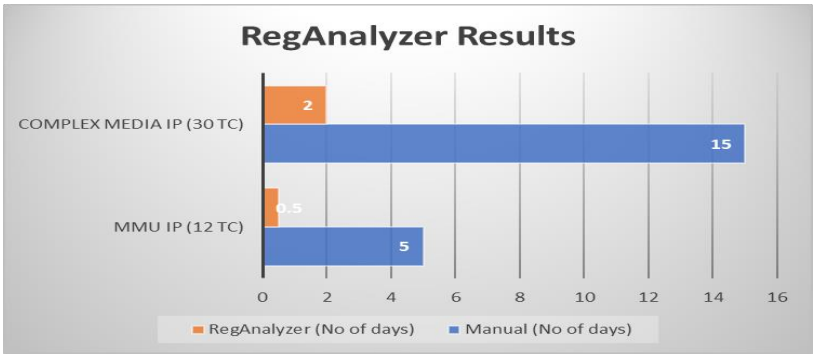


Figure 10. RegAnalyzer Results

V. CONCLUSION

RegAnalyzer is an effective GUI tool which will help in day-to-day verification activities, reduce the manual efforts drastically and save debug time. This tool will help during the design pre-silicon verification and post silicon validation to understand the register programming and its details. By using all the different features supported by this tool, verification and validation Engineers can make tremendous progress and reduce the overall debug time during Silicon-On-Chip development.

REFERENCES

[1] https://semiengineering.com/the-weather-report-2018-study-on-ic-asic-verification-trends/
[2] https://www.accellera.org/images/downloads/standards/ip-xact/IP-XACT_User_Guide_2018-02-16.pdf
[3] http://trace32.com/wiki/index.php/LDR_and_STR,_data_width
[4] https://www.design-reuse.com/articles/19895/ip-xact-xml.html