Mathematical Problems of Computer Science 39, 72--80, 2013.

Verification Environments for USB Controller

Grigor Y. Zargaryan

Institute for Informatics and Automation Problems of NAS of RA
e-mail: grigorzargaryan@gmail.com

Abstract

The complexity of electronic devices with the everyday growing requirements is constantly increasing. Sowtfare/Hardware (SW/HW) integration, validation and reducing time to market have become one of the major bottlenecks in the design and verification flow. This paper presents the main ways of design and verification flows with their advantages and disadvantages for USB 3.0 controller. It discusses the design flow with the combination of simulation and prototype-based design and presents a simulation-based verification and also two types of field-programmable gate array (FPGA) based verification environments with their advantages and disadvantages. The design done with this flow will enable system on a chip (SoC) designers to develop a high-quality USB 3.0 silicon solution to meet the growing market demands in a timely manner.

Keywords: universal serial bus, verification, simulation-based verification, FPGA-based verification.

1. Introduction

As electronic devices have already combined a lot of different functions, the market requires more and more new functionalities. However, it poses problematic issues such as a long development time and a hard design verification due to the increasing chip complexity. Another challenge also rises - to verify the complex design efficiently and timely under the situation the time-to-market is decreasing exponentially [1]. The dependencies of hardware and software result in an intricate relationship between the different company types. Fig. 1 shows the research results on failure types on the failing first silicone provided by Collett International [2].

Verification techniques can be classified into a simulation-based method and an emulation-based method [3]. In the simulation-based method, even if it has an advantage of being able to verify the design exactly and minutely, an excessively long simulation time is required. By using Register Transfer Level (RTL) HW models simulation, the verification times have increased to the level when they now take up to 70% of the device design time [4, 5]. In the emulation-based method, since it needs a certain emulation system such as a FPGA board, the board development time is added to the design verification time [6]. It is necessary to make a useful environment and to establish an efficient verification methodology for FPGA-based verification.

When RTL is largely verified and stable, the software development ramps up. It is split between OS support and porting, a low-level software development and a high-level application software

development. All the software development efforts consume 40% of the total cost for 27 months design[7]. When amortizing development and production cost onto expected sales, this project reaches break even after 34 months, i.e. seven months after the product launch but almost three years after the starting product development. The challenges in this example are that we have to predict nearly three years in advance what is going to be sold in high-volumes in order to specify our chip. How can this almost intolerable situation be made easier? The answer is to "start software sooner". If software development and validation started seven months earlier and subsequently the time to break even would have been reduced by five months. Additional revenue gain could be expected over the production volume due to submit to market design earlier than other similar products.

So it is extremely important to decide most appropriate development and verification flow to ensure the product success on the market and deliver high-quality, verified designs.

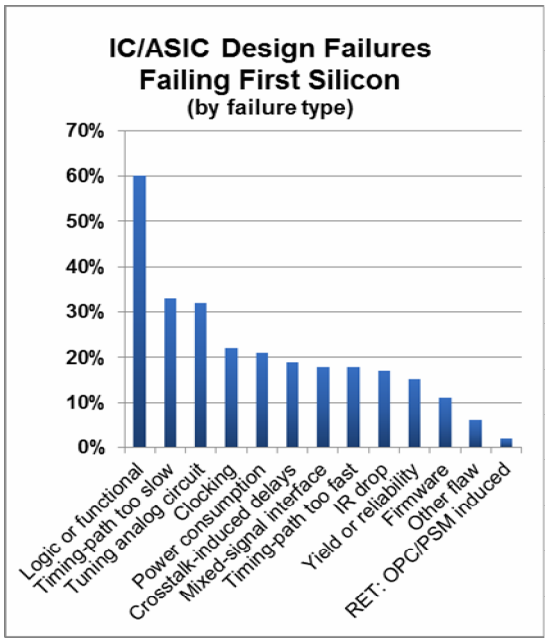

Figure 1. Failure types on the first silicone.

2. Simulation-Based Verification

The simulation-based verification model is one of the most popular and effective ways for functional verification. The simulation-based design requires more than just "a simulation tool". Depending on the complexity of the system, the design process may require many tools, many ways to link the tools together. The simulation environment that supports the design process should be flexible enough to disaggregate a complex system into any number of smaller pieces and conversely to aggregate independent objects into a complete representation of the system. It should, as much as possible, reduce the degree of complexity for the concept designer. This means that a graphical interface is essential, that energy couplings between the system components should be automatically and transparently handled, that the existing models should be reusable and that the rapid iteration of the design cycle and incremental refinement of the system should be supported on a group-wise basis [8].

Several approaches have been developed to reduce the simulation time and to increase the verification quality. One of the methods to reduce the device design time is the Transaction Level Modeling (TLM) [4]. It can be used for HW/SW modeling, co-design and co-verification. Another reason to use TLM model is the availability to start SW development at an early stage.

2.1 Simulation-Based Verification Environment

The verification environment is needed, which will allow us to run simulations and fix issues. USB host/device controller verification environment consists of a device under test (DUT), USB verification IP (VIP) and PCIe VIP (Fig. 2). It is possible to use other VIP instead of PCIe VIP, such as AXI/AHB/AMBA depends on what bus will be used next to the controller. If the design is implemented on the FPGA or on the ASIC, then DUT can be the top for this design, including USB controller and other necessary modules. Verification runs as follows: each transaction is a combination of a request and response. VIP starts the transaction by sending a request and waiting for an appropriate response. Simulation tests pass when all the required requests are sent and the responses are received.. VIP generating test vectors using SystemC, which allows to short verification time. Figure 3 provides a part of SystemC code which compares the data transferred between the device and the host using USB simulation environment. After running simulation the .vpd file can be used for debugging. The DVE tool provides graphical user interface which allows performing debugging in effective ways. Figure 4 shows ssrxp, ssrxn, sstxp, sstxn and ssclk states of the signals after performing simulation. The ssrxp, ssrxn, sstxp and sstxn signals are external connections and responsible for date transfer on Super Speed mode.

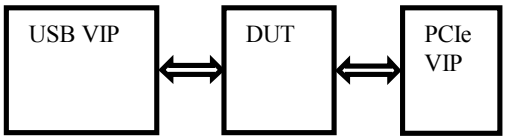

Figure 2. USB verification environment.

```
$display(" Data comparison(Tx and Rx Buffers) after transfer is done... @", $time);
  first = 0;
  for(j=0; j < dut_number_of_trbs; j=j+1) begin
    for(i=0; i < dut_bytes_per_trb/4; i=i+1) begin
      tmp_data32  = start_data + i*step;
      k = (j*1280) + i;
      tmp_data2   = {32'h0, mem[3+4*k], mem[2+4*k], mem[1+4*k], mem[4*k]};
      if(first == 0)
        $display("First Location-A: Rx buffer data= %h; Tx buffer data= %h",
tmp_data2[31:0], tmp_data32, $time);
      first = 1;
      if(tmp_data2[31:0] !== tmp_data32) begin
        fail = 1;
        $display("Data Mismatch; expected= %h; received= %h @%t", tmp_data32,
tmp_data2[31:0], $time);
      end
    end
  end
 end
if(fail==0)
  print_banner(" Comparison Passed ");
 else
  print_banner(" Comparison Failed ");
end
```

Figure 3. System C code performing comparision of transferred data.

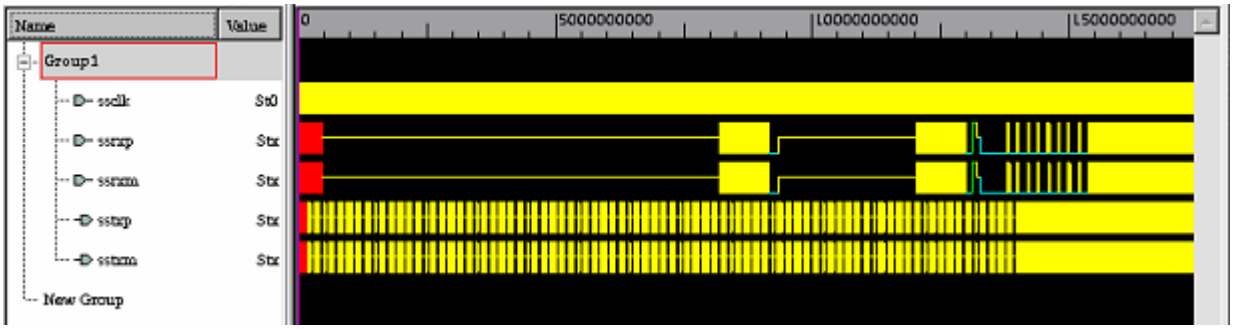

Figure 4. Simulation diagrams

3. FPGA-Based Verification

FPGA-Based Prototyping accelerates the creation of an ASIC prototype with high-speed hardware prototyping systems including a software flow for the conversion of ASIC RTL into one or more FPGA ICs. FPGA-based prototypes provide cycle-accurate, high-performance execution and real world interface connectivity prior to tape-out of test chips.

In the effort to reduce Time-To-Market (TTM) engineering organizations continue to seek ways to develop hardware and software in parallel. Advanced ASIC prototyping techniques enable a more parallel development methodology. And firms, which have achieved more concurrent engineering practices, have not only reduced the time to product introduction, but additionally reduced Product Support & Maintenance effort during the product's Time in Market due to higher quality.

The sooner the real Software Development begins, the more feasible it will be to make progress on the Integration & Test, and validation phases prior to the tape-out milestone (Fig. 5).

From the aspects of debugging and control capabilities, the virtual platforms or any simulation allow much easier ways than FPGAs. But on the other hand, FPGA allows much debugging and control capabilities than the actual silicon provides when available. To allow debugging on FPGA boards before running synthesis it is necessary to define which signals will be used for debugging. Also additional tools are required to grab debugging signals from FPGA platform. If some additional wires or signal are needed for debugging which are not defined before the synthesis, it will be necessary to define and rerun the synthesis again. Also it is recommended to start FPAG implementation after RTL verification has stabilized due to the efforts of mapping the RTL to FPGA-based prototype. For the same reason it is not useful for hardware/software co-development. Prototyping provides powerful methods for validating the design of hardware and software in models. FPGA-base prototyping is specifically useful during the hardware and software integration.

Nowadays FPGA technologies allow high density, high speed, broad bandwidth, low-voltage, low-power and low cost. There are built-in IP cores, which can extend the application area and shorten the cycle of R&D. More functional cores like networking, audio, video and image can be integrated into a single FPGA chip.

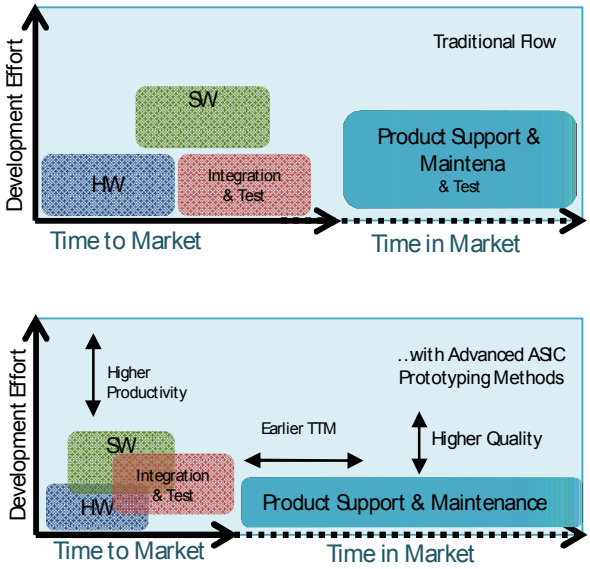

Figure 5. Terms Reduce Time-To-Market

3.1 FPGA Implementation

The FPGA-based verification environment consists of PC, PCIe cards, FPGA board and USB physical layer (PHY) (Fig. 6). An appropriate OS will be loaded on the PC with the controller driver and high level applications. The PCIe bus will make a connection between PC and FPGA board. FPGA board is connected to USB PHY with a parallel interface, such as PIPE3, ULPI and UTMI. USB PHY layer contains an analog receiver, transceivers and convert sequential data into parallel. This type of environment will allow easy software debugging.

Figures 7 and 8 present host's and device's logical components. Host includes the following: USB Host Controller, Aggregate USB System Software (USB driver, host controller driver and host software), Client. Device includes the following: USB bus interface, USB logical device, Function[9]. On the market there are few companies that provide tracers for packet level debugging. There are PCIe and USB tracers which can be used for more effective debugging. Figure 9 shows an example of USB 3.0 trace recorded on this type of environment.

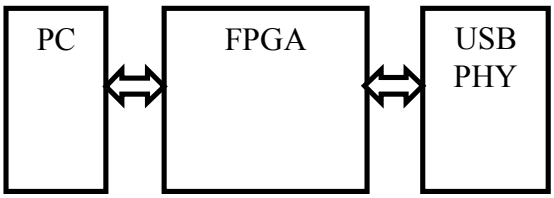

Figure 6. USB controller implementation

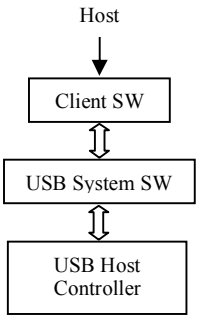

Figure 7. USB Host's logical components

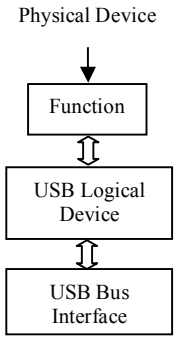

Figure 8. USB device's logical components

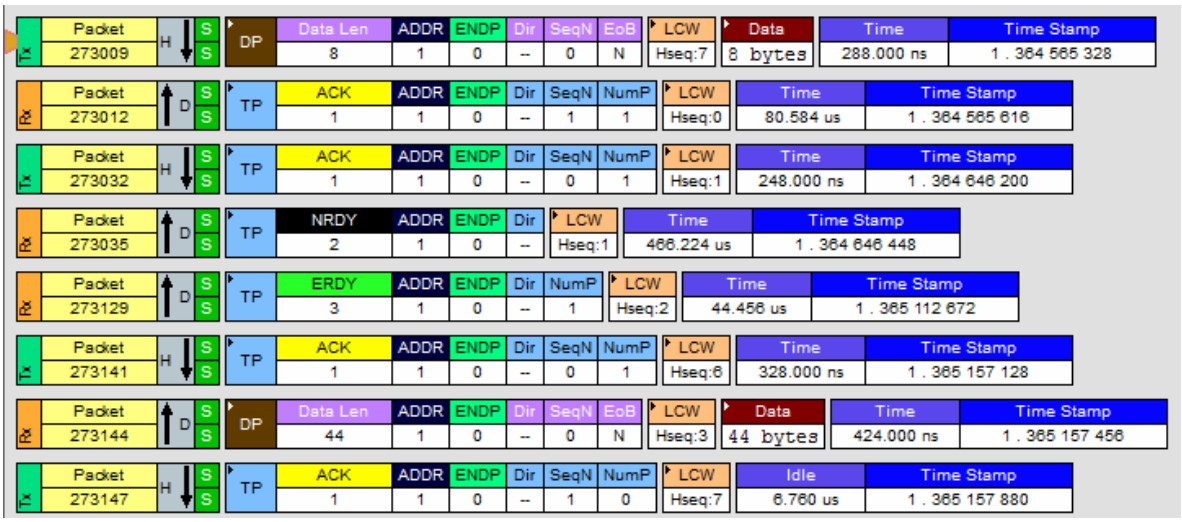

Figure 9. USB 3.0 trace example.

4. FPGA-Based Embedded Systems

Embedded system often refers to the non-PC systems which combine hardware and software design. In general, it contains embedded micro-processor (8-bit, 16-bit or 32 bit), storage and peripherals, embedded OS (real-time and multi-task) and applications (Fig. 10).

Embedded systems have some characteristics which differ from other computing systems [10].

- Small system kernel.
- Specific-functioned.
- Real-time OS.

In terms of embedded hardware, its core component is the embedded microprocessor. At present there are over 1,000 kinds of embedded processors in the world and the popular architectures are more than thirty, in which Intel MCS-8051 is ever the overwhelming majority. In recent years the small volume, high performance and low power consumption become dominant factors of embedded system design considerations. The professional intellectual property (IP) core providers like ARM, MIPS Corps. offer high-quality embedded cores to semiconductor manufacturers, by which all kinds of chips on different devices applied to diverse areas, are widely produced.

4.1 FPGA-Based Embedded Environment

Embedded design flow combine embedded SW flow and FPGA HW flow. The hardware design flow consists of standard FPGA design steps such as design entry, simulation, synthesis and implementation. The software design flow consists of C code, C/C++ compilation to linker and debugger. Generated HW binary file for FPGA configuration and SW code written into the board through JTAG.

This type of setup verification environment also requires USB PHY, and consists of target board and USB PHY (Fig. 11).

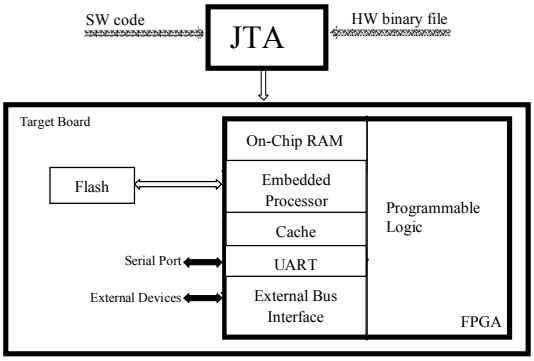

Figure 10. Design diagram of FPGA-based embedded system.

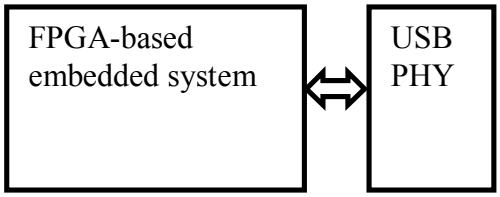

Figure 11. FPGA-based embedded system implementation.

5. Conclusion

Simulation-based verification can be used to start developing USB controller. Developing SystemC level models and parallel workaround on HW and SW can allow shorter time to market. SystemC models usage allows fast simulation time. After RTL has stabilized FPGA-based or FPGA-based embedded system can be created. Prototyping can help most in the following SW Validation and Integration tasks: OS configuration & installing, kernel space debugging, on-chip debugging, user space debugging, unit testing, system testing, field diagnostics and lab diagnostics.

FPGA-based prototypes in particular provide the most help in the highlighted area enabling:

- Physical-layer interface compatibility checking
- At-Speed Debug
- Regression Testing
- Multi-Core Integration
- In-Field Tests

And finally prototyping USB controller allows testing SW and HW with real world USB PHY and with real world USB devices. Testing with real world devices and real speed will allow silicone success on first tape out. It is very hard to imagine more useful environment than the real world testing environment.

Real world testing at USB 3.0 speeds helps to verify the architecture, such as memory management and interoperability tests for USB 3.0 standard compliance. Finally, the FPGA validation platform can also be used for USB Implementers Forum certification of a prototype design and Windows Hardware Certification Kit by Microsoft. New driver stacks are required to handle the faster USB 3.0 speeds, and simply extending USB 2.0 architectures to support USB 3.0. Universality of the USB protocol requires that hosts are tested with hundreds of USB 2.0 devices and all available USB 3.0 devices.

Real world prototype can be present in many technical exhibitions. Also it can be given to customers to try if it meets their needs, to try different configurations of RTL, different modes, etc.

References

[1] E. Jimenez, “Challenges in system on chip verification”, *International Workshop on Microprocessor Test and Verification*, pp.52-60, 2006.
[2] Source: Collett International Research, Inc.
[3] C. Pixley, et al., “Functional verification 2003: technology, tools and methodology”, *International Conference on ASIC*, vol.1, pp.1-5, 2003.
[4] S. Swan, “SystemC transaction level modelsand RTL verification”, *Proc. 43rd ACM/IEEE Design Automation Conference*, pp. 90-92, 2006.
[5] S. Tasiran and K. Keutzer, "Coverage metrics for functional validation of hardware designs", *IEEE Design & Test of Computers*, vol. 18, no. 4, pp. 36-45, 2001.
[6] Y. Lin, et al., “Versatile PC/FPGA-based verification/Fast prototyping Platform with multimedia applications”, *IEEE Transactions on Instrumentation and Measurement*, vol.2, pp.1490-1495, 2007.
[7] A.Doug, L. Austin, *FPGA-Based Prototyping Methodology Manual*, Published by Synopsys, inc., Mountain View, CA, USA, 2011.
[8] R. A. Dougal, “Design tools for electric ship systems”, *IEEE Electric Ship Technologies Symposium*, pp. 8-11, Philadelphia, PA, July 2005.
[9] USB 2.0 Specification, April 27, 2000, www.usb.org.
[10] F. Vahid and T. Givargis, *Embedded System Design – a Unified Hardware/SoftwareIintroduction*, John Wiley & Sons, Inc., pp. 1.1-1.2, 2002.

Submitted 20.12.2012, accepted 21.02.2013.

Համապիտանի հաջորդական դողի ղեկավարող հանգույցի ստուգման միջավայրերը

Գ. Զարգարյան

Ամփոփում

Էլեկտրոնային սարքավորումների բարդությունը և նրանց վրա դրված պահանջները օրեցօր աճում են: Ապարատա-ծրագրային համադրումը, ստուգումը և շուկա դուրս գալու հրատապությունը նախագծման փուլում դարձել են կարևորագույն պահանջներ:

Աշխատանքում ներկայացված են համապիտանի հաջորդական դողի (ՀՀԴ) ղեկավարող հանգույցի նախագծման և ստուգման փուլերի հիմնական ուղիները՝ իրենց առավելություններով ու թերություններով:

Կախված նախագծման և ստուգման առկա վիճակից՝ տարբեր մոդելներ ունեն տարբեր արդյունավետություն: Հաշվի առնելով ստորև ներկայացվածը՝ կարելի է սեղմ ժամկետներում նախագծել բարձրակարգ ՀՀԴ-ի ղեկավարող հանգույց:

Среда проверки управляющего узла универсальной последовательной шины

Г. Заргарян

Аннотация

Сложность электронного оборудования и предьявляемые к нему требования растут с каждым днем. Аппаратно-программное сопоставление, проверка и актуальность выхода на рынок стали важнейшими составляющими на этапе его проектирования.

В работе представлены основные пути этапа проектирования и проверки управляющего узла универсальной последовательной шины (УПШ) со всеми своими преимуществами и недостатками.

В зависимости от способа проектирования и проверки, различные модели имеют различную эффективность. С учетом нижеизложенного можно в сжатые сроки проектировать высококачественные управляющие узлы УПШ.