Improving the Design Process of VLSI Circuits by Means of a Hardware Debugging System: UNSHADES-1 Framework

M.A. Aguirre, J. Tombs, A. Torralba and L.G. Franquelo
Escuela Superior de Ingenieros. Universidad de Sevilla
Avda. Camino de los Descubrimientos s/n,
4109 Sevilla (SPAIN)
aguirre@gte.esi.us.es, jon@gte.esi.us.es,, torralba@ieee.org, leopoldo@ieee.org

***Abstract* – Due to the increase in size and complexity of VLSI integrated circuits, new design tools are becoming needed. Telecommunications and Electronic Industry demand designs that integrate intensive digital signal processing blocks and complex control tasks. Rapid Prototyping techniques introduce a new stage into the design flow that overcome the drawbacks of simulation stage and shorten design times. Advanced FPGAs can host the design for its emulation and can run inserted into the final system. The benefits of their use go beyond the simple rapid prototyping approach, and are able to provide additional information and other useful tasks that will be presented in this paper.**

I. INTRODUCTION

Due to the increasing complexity of many electronic projects, the design flow of VLSI integrated circuits has to be completed [1] with the use of some prototyping or emulation techniques. Figure 1 show the complete design flow including the prototype phase. The traditional simulation stage isn't enough, because stimuli generation does not always represents all the run-time situations and, in most of cases, it's almost impossible to generate enough data to activate all cases in the deepest design blocks inside the design hierarchy tree. Stimuli are usually generated using well controlled models of the system. However, if the prototype is essayed using the physical elements of the final system, non controlled (in terms of modelling) situations can be introduced taking into account the real external components. At the same time internal scope techniques have to be introduced to obtain information about what is actually happening inside the system.

The prototype is typically supported by means of an advanced SRAM-FPGA [4] these can normally work at high clock frequencies. The main advantages of using FPGAs for this purpose are basically: First, there's an unlimited number of configuration cycles, that allows the verification of multiple versions of the circuit netlist, and second the emulated version can be produced with extra circuitry and pins for providing extra execution-time information.

Advanced FPGA's are SRAM based programmable circuits that can host large digital circuits with a reasonable device behaviour. These devices are typically built with other features very useful for building the prototyping system itself, such as many electrical iopin standards, JTAG ports, clock skew compensation, general purpose multipliers, on-chip memory blocks… etc.

The concept of a Hardware Debugger [6-7] is more ample than a simple prototyping system . Using an advanced FPGA, it not only can emulate the final circuit but it can provide information about how the circuit is working during execution, and eventually provide more control on the running device that can be exploited for a better analysis. Basically the added capabilities are: cycle by cycle stepping of the entire circuit, extract run-time information for internal scope, and, eventually, produce modifications in the internal registers.

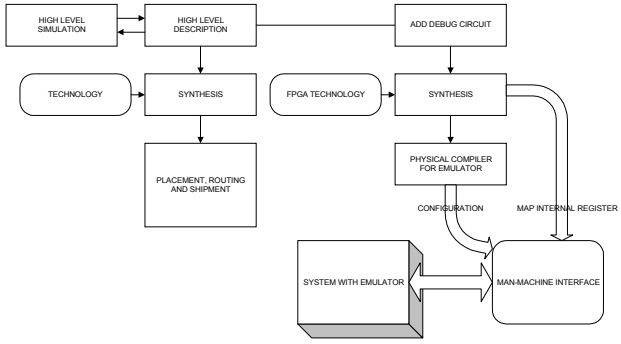

Fig 1**.** Modified VLSI Design Flow

Most FPGA foundries offer debugging packages that support recording internal signals into a synthesised memory, like an internal logic analyser. These products are Chipscope from Xilinx [9-10] and SignalTap from ALTERA. Basically they consist of a circuit for the run-time condition detection that triggers the signal capture process. Some memory resources are reserved for recording the preselected signals. All of these resources that have to be included in the same circuit netlist. The size and behaviour of the placed and routed circuit depends strongly of how many signals and clock cycles are going to be to be recorded and the presence of those elements strongly affects to the behaviour of the emulated circuit. The extra size and the modified performance of the circuit is called the *overhead* of the debugging system.

This paper presents UNSHADES-1, a different approach based on VIRTEX FPGAs where some special features that are exclusive to this FPGA are exploited for constructing a hardware debugger system. UNSHADES-1 means UNiversity of Sevilla HArdware Debugging System and is based on the platform HADES-1 [2,5]. The complete system, software and hardware are inserted into the Xilinx

standard design packages, Foundation and Alliance. This paper is organised as follows: Section II shows an overview of the VIRTEX properties that are useful for UNSHADES system. Section III describes how to execute the step by step system. In section IV a two wires system for controlling the debugger is presented and Section V y VI briefly present some ideas about the software and finally section VII will show the conclusions.

II. HARDWARE DEBUGGING SYSTEM

UNSHADES-1 is a system independent hardware debugger where the designed circuit is substituted in the final system by one advanced FPGA, a device taken from VIRTEX series from Xilinx. A computer supports the interface man-machine. UNSHADES board [5] is basically a smart link between the FPGA and the computer, where it can download the configuration to the FPGA and can read and write partial information of that configuration. This features play an important role in our system because it means that FPGA can be explored and manipulated in order to obtain inner view and control of the device during execution.

VIRTEX FPGAs [7] form a family of devices from Xilinx that support large amount of system gates with high level circuit performance. The next paragraphs will highlight the most interesting features of these FPGAs for our purpose.

Using the SelectMap port Virtex can be easily, configured, read and written, totally or partially. The information can be moved as 8-bits wide data at a high speed (until 60Mb/s) during the partial reading and writing. This allows fast selecting of the information to read or modify.

Secondly, there's a single-clock cycle process for saving the internal state (the contents of the all the Flip-flops) for external reading: the CAPTURE_VIRTEX macro. After the execution of the macro the saved state can be uploaded to the host and the information about the actual register contents can be linked with their designer's names, given during design time thus making the recorded information comprehensive to the user. The state information is copied to special dedicated memory cells that are not included in the common resources of the FPGA and this process doesn't contributes to the overhead of the UNSHADES-1 system.

Finally, all the Virtex flip-flops have a *clock enable* input for enabling synchronous changes. This input can be safely manipulated to control the execution of the design instead of using signals that control on the clock wires themselves.

III. STEP BY STEP EXECUTION

Stopping a system clock is not easy in large FPGAs, because clocks are compensated using phase compensation techniques that need several clock cycles to be settled and cannot be resumed. Before the system stops, a debug condition has to be reached, this is done with a pre-programmed condition that watches the running system. When the condition is satisfied the system freezes the circuit. Instead of stopping the system clock the Clock Enable input of all the flip-flops can be used to freeze the circuit. In this case, the circuit can be frozen partially. We have used this input for obtaining a full control of the execution process, free of glitches.

With an extra external input, a simple state machine can control the clock enable input of the flip-flops. At every rising edge of this signal a finite state machine enables the system a single clock cycle and activate the CAPTURE macro. After this, the host can upload the new information.

For introducing a step-by-step execution mechanism into the design netlist the following steps have to be performed:

1. Attach the line *freeze* to all flip-flops that are going to be frozen. This can be done introducing the flowing code into the high level description code (for example, in VHDL) [9].

```
if(clk'event) and clk='1')then
        if(freeze='1') then
        .....
        end if;
end if;
```

2. This line will be controlled by a state machine that de-asserts it when the freeze condition is taken and produces the CAPTURE_VIRTEX procedure.
3. The PC uploads the information containing the circuit state.
4. Using an external line the PC activates the circuit for a single clock cycle. The next clock cycle a new CAPTURE_VIRTEX procedure is activated.
5. Again the PC uploads the information.

Steps 4 and 5 can be repeated as many times as desired, and, at every step the register contents can be recorded in a file with a valid format suitable for being displayed in a waveform viewer.

Figure 2 depicts an block scheme of how the described system works where CAPTURE_VIRTEX macro is activated when the clock enable is deasserted.

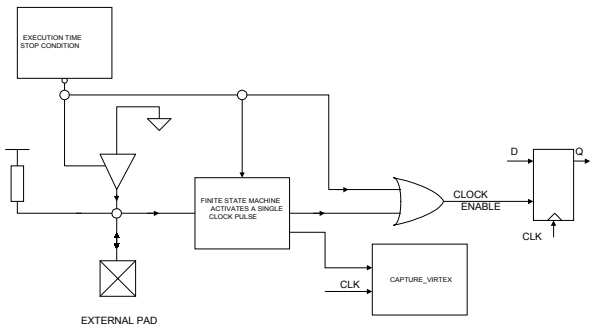

Fig 2. Step by step machine

The number of consumed resources for this circuit is negligible, less than one hundred system gates and it can be shown that there's no effect on the circuit behaviour, because

the run-time condition for activating the capture system is a simple comparator.

The pullup resistor provides a bi-directional character to the external pad. It indicates that the circuit is running when low, but after the condition, it goes to resistive high ready for management by the step by step system.

IV. TWO WIRES METHOD

This section will explain how the UNSHADES-1 emulator informs to the control computer about the detection of a particular event. We have introduced a two wires method. Thus, our debugger will have an overhead of two pin for its own control. These two lines are:

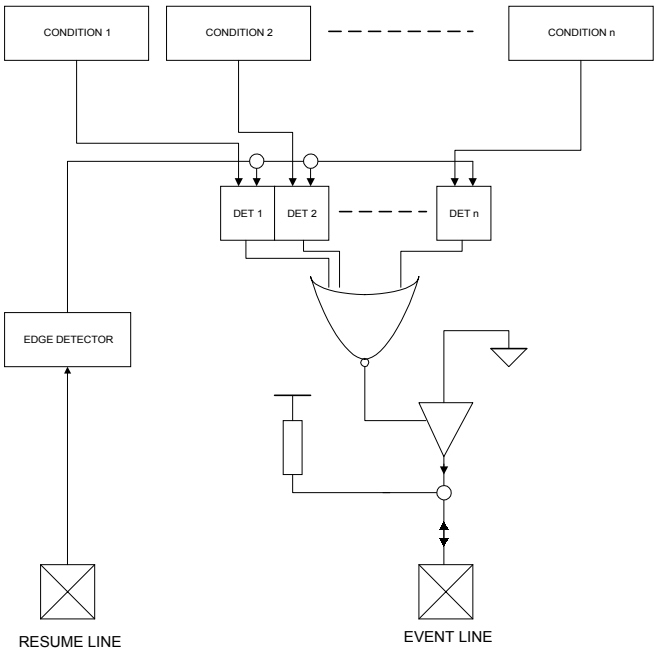

Fig 3. Two wires control system

- Event line: This line is shared with the external pin used in the step by step method. It is an open collector line (as explained before). When the circuit is working, this line is deasserted. All the programmed events are or-ed to this line. When any event is asserted then after a readback process the event can be identified.
- Resume line: This line will reactivate the circuit after an event has been detected, or erase the run-time event.

Figure 3 shows a scheme of how the system is built. It can be seen that both systems step by step and two-wires controller share the *Event Line*, due to the three state character.

After an event the emulator has to get the necessary information for the user. The method is associated to the Xilinx macro CAPTURE_VIRTEX, where the circuit state is copied in one clock cycle to an special configuration memory. After this trigger the system can be stopped (and stay frozen) or continue working. In the first case the debugging process loses the synchronization with the external system but no clock cycles are lost and in the second the emulation gives the information of a single snapshot of the system.

Note that the debugging condition is not restricted to a single condition. The user can fix a priori as many conditions as desired. Main advantage of this method is that the actual debug condition that triggers the system is identified in the same way as the internal registers and signals.

V. INSERTION OF THE DUBUGGING CIRCUIT

Debug circuit can be automatically inserted into the emulator without being resynthesized. The designer takes decisions about which wires or registers have to be watched to define the trigger condition for the first snapshot. Note that due to the pure parallel nature of hardware all conditions are watched simultaneously. One comparator has to be inserted per event. This extra circuit can be inserted to the high level code, but some design time can be improved if the circuit is inserted to the structural netlist (VHDL, EDIF,...) at the previous stage of physical implementation. The main advantage is that the designer debugs the final netlist, and doesn't have to re-synthesise the complete design, which is more or less 40% of the total synthesis time. This task can be done automatically using a tool that is being developed in our department.

VI. UNSHADES SOFTWARE

The Man-machine interface is an essential part of a hardware debugger. In the same way that information is presented, performance superior to the complete system is obtained as the system can be inspected and manipulated. In this way the system has to be fast and effective. We take advantage with the partial reconfiguration techniques of VIRTEX device. The reconfiguration time is then very fast because only a small part of the FPGA has to be inspected or configured. For example, a frame (the minimum amount of information transferred) of VIRTEX 50 can be inspected in less than 10 microseconds.

In order to display step-by-step information, a standard format (vcd) database is produced and a waveform viewer displays the evolution of the emulated circuit.

Moreover, if we have the system in a frozen state some manipulations can be produced in the state, allowing the modification of the values of the flip-flop contents. This novel idea is also being developed in the new software tool.

VI. CONCLUSIONS

The idea of debugging digital circuits in run-time is attractive. Due to the improvements of the FPGA's the circuit can be emulated and inserted into the final system previously to being sent to the foundry. FPGA's provide more features than expected and can be instrumented for the design process. In our Department we have developed UNSHADES-1, a new tool for debug digital circuits. With the insertion of an extra logic the circuit can be emulated in a controlled way without being affected of the control circuit itself. UNSHADES system present an alternative non exclusive to the commercial debugging systems.

VII. REFERENCES

[1] Dollas, A. Kanopoulos, N. "Reducing Time to Market Through Rapind Prototyping". IEEEComputer, Feb. 1995.

[2]A.Burst, B.Spitzer, M.Wolff, K.D. Müller-Glasser. "On Code Generation of Rapid Prototyping Unsing CDFI". OOPSLA'98, Vancouver, CA

[3] Hutchings B., Nelson B. and Wirthlin M.J.. "Designing and Debugging Custom Computing Applications". IEEE Design & Test of Computers. Jan-March 2000. pp 20-28.

[4] Koch A. "A Comprehensive Prototyping-Platform for Hardware-Software Codesign". 11th IEEE Internation Workshop on Rapid System Prototyping. Paris Jun-2000. pp78-82.

[5] M.A. Aguirre, J. Tombs, A. Torralba, L.G. Franquelo "HADES-1: A rapid prototyping environment based on advances FPGA´s". Proceedings of the Design of Circuits and Integrated Systems, DCSI'01 Oporto 2001.

[6] Timothy Wheeler, Paul Graham, Brent Nelson, and Brad Hutchings, ``Using design-level scan to improve FPGA design observability and controllability for functional verification," in Proceedings of the Eleventh International Workshop on Field Programmable Logic and Applications, pp. TBA, Belfast, Northern Ireland, August 2001.

[7] Paul Graham, Brent Nelson, and Brad Hutchings, ``Instrumenting Bitstreams for Debugging FPGA Circuits," in J. Arnold and K. Pocek, editors, Proceedings of the Ninth Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp. TBA, Rohnert Park, CA, April 2001.

[8] Blind for the reviewing process

[9] Xilinx. Co."The Programmable Logic Data Book". 2001.

[10] Xess Co. XSV Board V1.0 Manual. March 2000