2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE OCTOBER 26-27, 2021

Advance Approach for Formal Verification of Configurable Pulse Width Modulation Controller

Sumit K. Kulshreshtha, Intel Technology India Pvt Ltd, Bengaluru, India
(sumit.kumar.kulshreshtha@intel.com)
Raghavendra J N, Intel Technology India Pvt Ltd, Bengaluru, India
(Raghavendra.Jn@intel.com)

***Abstract*— Artificial Intelligence (AI) based systems require intelligent and configurable hardware which can support AI applications. This leads to the development of smart hardware designs which can support a wide variety of AI applications. One of such designs is a smart Configurable Pulse Width Modulation (CPWM) controller which is very critical for proper system functionality as it controls LASER falling on human eye. With latest competitive trends, it becomes essential to deliver such designs in much lesser time to have better time to market (TTM). One of the major time-consuming process is verification of these hardware designs. This paper describes usage of advance and effective formal verification techniques to verify such hardware designs in much lesser time with higher efficiency.**

Keywords*— *Formal verification, overlapping checkers, coverage, exhaustive verification.

I. INTRODUCTION

Rising demand for Artificial Intelligence (AI) and Machine Learning (ML) based systems have revolutionized the VLSI industry. These AI/ML based systems require complex system on chips (SoCs) and these complex SoCs need innovative configurable hardware designs to support the required functionality. Few of the design blocks are very critical for proper functioning of a system and need to consider the human safety as well based on the application area. One of such complex hardware designs is a Configurable Pulse Width Modulation (CPWM) controller. A CPWM controller can be used in many critical applications such as flash triggering, laser power control, sensor synchronization and scheduling. One of the possible use cases of CPWM controller is controlling LASER falling on human eye. All these applications are critical for human safety and require the designs to be working exactly as expected without compromise else even a small design error may turn fatal. Hence it becomes essential to make sure that these designs have been tested for all the possible scenario and have no bug. This poses two different verification challenges as listed below:

Exhaustive verification – Due to the critical applications of CPWM controller, it becomes essential to verify such designs exhaustively for all possible configurations.

Verification time – Now-a-days, the product development time is very short due to huge competitions. Due to smaller product development cycle, the time allotted for verification is very less.

Using simulation-based verification and exercising directed or constrained random tests is not feasible due to large configuration space. Covering all the possible scenario with directed or constrained random tests is not possible when the configuration space is large. Using random tests is also not a good approach due to limited time to market and random tests don't guarantee for complete coverage in limited time. Hence due to very high degree of configurability and limited time to market, it is practically impossible to completely verify CPWM controller using simulation-based verification approach.

In order to overcome above limitations of simulation-based verification, Formal Verification (FV) can be adopted which provides exhaustive verification. But tradition formal verification suffers a problem of convergence for complex designs. Dealing with FV convergence is a time-consuming process. This paper describes a proven formal verification approach to exhaustively verify a complex configurable design. Formal verification can

effectively be used to verify such designs for all the possible configurations and for all the data values in much lesser time.

II. DESIGN OVERVIEW

The design considered in this paper is a configurable pulse width modulation controller as shown in Figure 1. It is having an input interface for design configuration and various input events which trigger the output waveforms as per the configurations. The input events go through a cross bar and based on the configuration; they trigger different waveform controllers. These waveform controllers act on these triggers based on their current state. The output of each waveform controller is connected to different output pins. The window qualifier controls the shape or the duty cycle of the output waveform. The output pins drive the General-Purpose Input Output (GPIO) pins of the SoC. These GPIOs drive LASER beams which may be directed to human eye for few of the applications. Based on the incoming event and current waveform controller state, interrupt controller generates different types of interrupts. Details of various internal design blocks are described as below.

Configuration Registers: This is the system configuration space which controls the design functionality and update the system behavior based on the inputs on configuration bus.

XBAR: This block does the routing of input events based on the design configuration. It redirects input events to various waveform controllers.

Waveform Controller: This design part receives a waveform trigger from XBAR and based on this trigger type and system configuration, it generates an output waveform. The duty cycle, pulse width and number of cycles are calculated based on the system configuration.

Int Ctrl: This is an interrupt generating block. It generates an interrupt based on the waveform trigger from xbar and the waveform controller state. The interrupt controller works as per the system configuration.

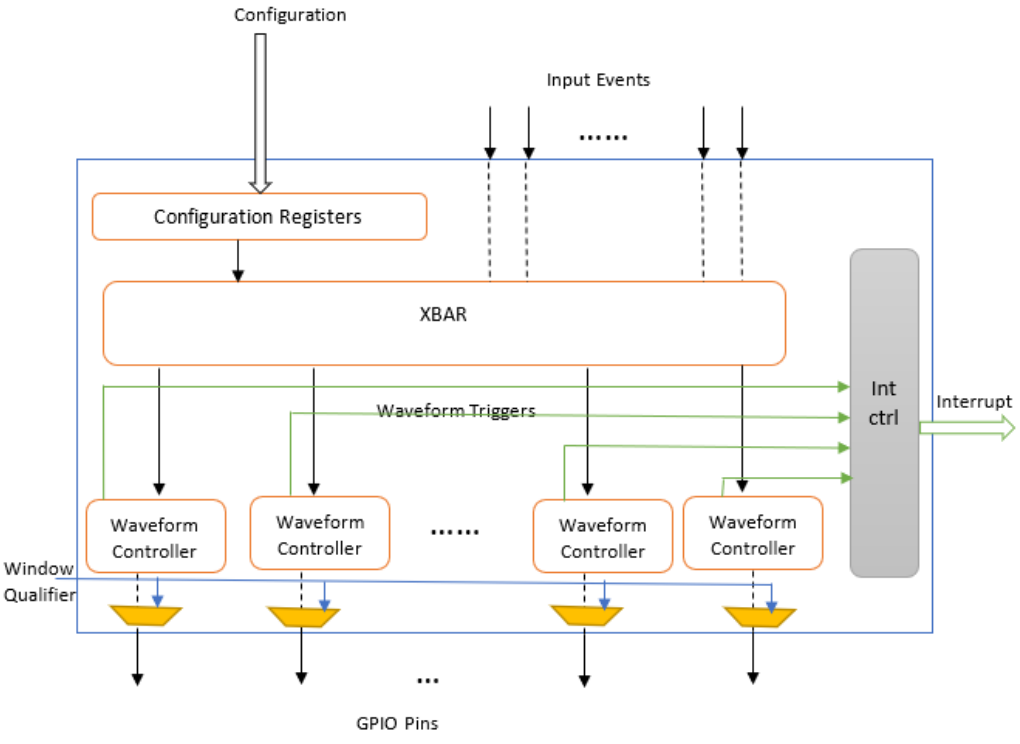

Figure 1: Configurable Pulse Width Modulation Controllers

III. CPWM CONTROLLER VERIFICATION STRATEGY

This section explains various techniques which were used to reduce the verification time and provided complete coverage. The fundamental verification strategy used was Formal Property Verification (FPV). We used FPV along with few of the latest techniques and closed the verification early. The techniques adopted in verification of this design are explained in following sub sections.

A. *One cycle design configuration:*

This is a method of reducing the time consumed in design configuration. A design is configured by writing to the configuration registers using a standard (or customized) bus protocol. Writing to these registers generally takes few cycles. If we want to configure the design in different ways, we will end up wasting many cycles. This section shows how we can save these cycles and accelerate the verification or a property convergence in lesser time using a method called “one cycle design configuration”. The technique of “one cycle design configuration” helps in three different aspects as

- Reduce the design complexity.
- Explore all the possible configurations in much lesser time.
- No need of constraints to drive correct values on the configuration bus.

Typically, the configuration registers are pre verified shared devices. If they are new designs and big enough, they can be taken out and verified separately. In the given Design Under Test (DUT), we black boxed the configuration registers and put constraints at their outputs to avoid any illegal stimuli. This saved a lot of time which is otherwise used in writing or updating these registers. With black boxed version, the design complexity was reduced. These registers were left to assume any random (conditionally constant) values so that the design is tested or verified for all possible values of system configurations in zero time. This avoids effort in writing set of constraints to drive correct values on the configuration bus. As soon as the system comes out of reset the design is ready to be tested for a specific configuration set.

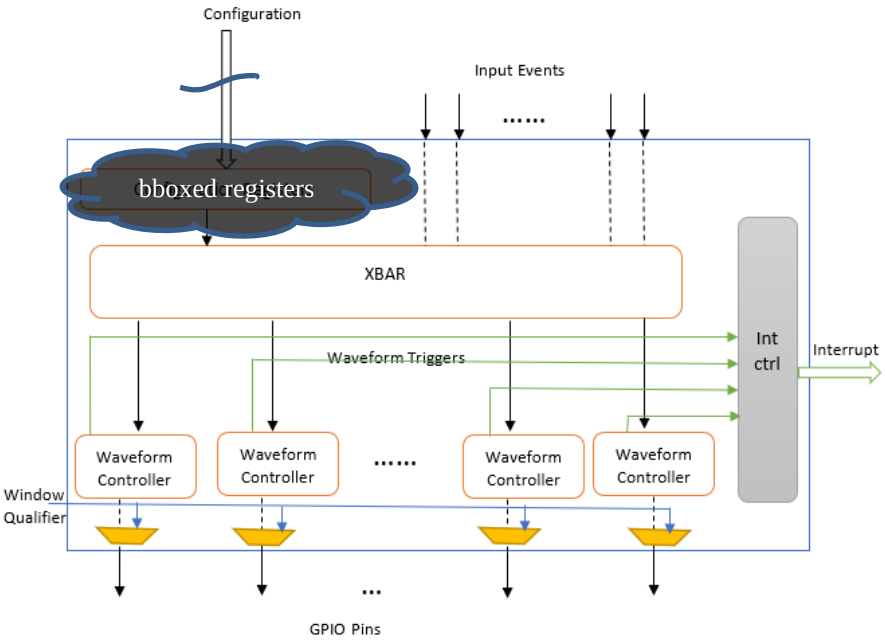

Figure 2: Black boxed CPWM Controller

*B. **Using overlapping checkers:***

The CPWM controller was a complex design hence getting full convergence for end-to-end checkers was not so easy. We had a checker C1 which checks if an input event generates expected waveform at the GPIO output pin or not. After long runs, the checker C1 was inconclusive and the bound was also not good. After that we applied various deep bug hunting methods still the bounds were not very good. Hence, we decided to use the overlapping checkers concept [1]. The CPWM controller was logically partitioned into two smaller parts and then end to end checkers were written for each of these partitions as shown in Figure 3. First part covers the design from primary input events to waveform triggers. We wrote an end to end checker C2 for this part which checks if input events cause correct waveform triggers or not. Second part covers the design from waveform triggers to GPIO outputs. An end to end checker C3 was written for this part which checks if these waveform triggers are generating the expected waveforms at the GPIO pins or not. Both the checkers C2 and C3 were converged which provided much more confidence for signoff than single checker C1 for end to end functionality.

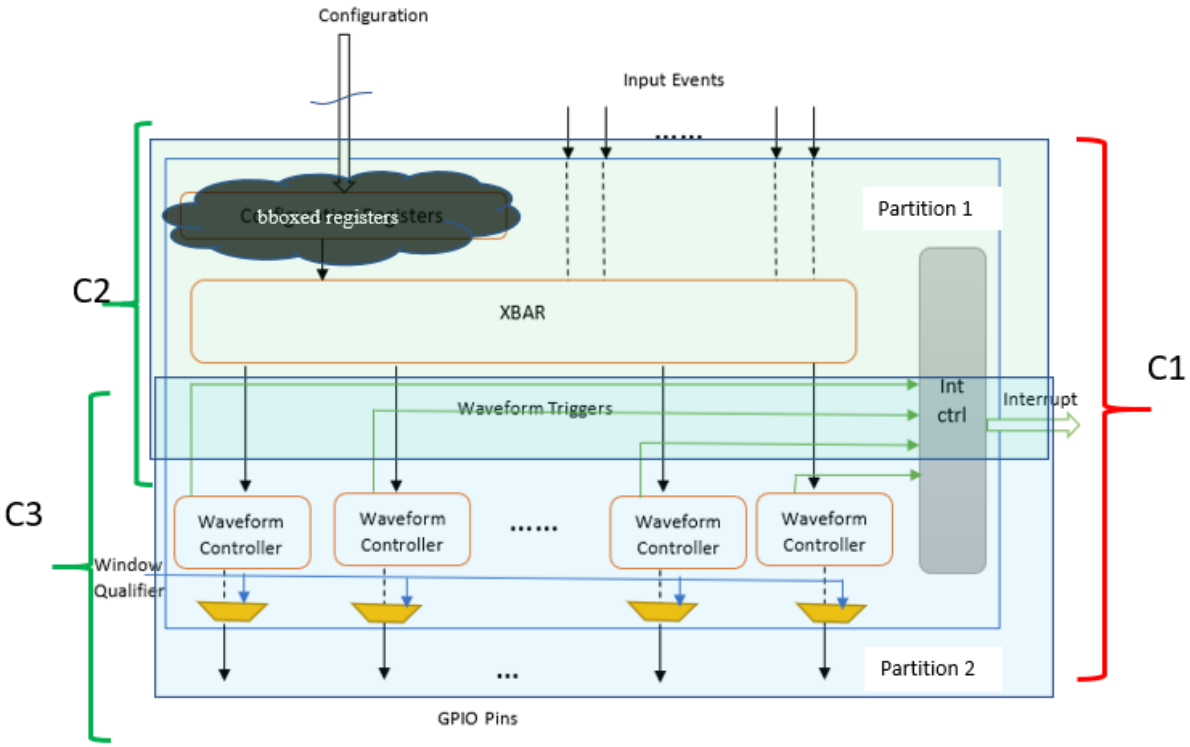

Figure 3: Overlapping Checkers with logical partitioning of CPWM Controller

C. Overall Verification Strategy

The expected time for simulation-based verification of this design was 20 days and the expected time using FPV was 10 days. We planned in such a way that the overall verification was completed in less than the expected time. While developing the verification plan for CPWM controller, we focused on the three areas of formal property verification Constraints, Checkers, Coverage.

Constraints: In the beginning, there was a need to write constraints to drive correct set of values on the configuration bus using a standard bus protocol which was a time-consuming process, and which could also increase

the complexity. Hence, we decided the zero-cycle design configuration method as explained above. With this method, there was no need to writing this constraint set. Instead we just had very few (4) constraints to have correct values in different registers. We left the input events unconstrained to have all the possible scenario.

Checkers: The checkers were divided into two different parts of the DUT. First one is local checkers and the second one is end to end checkers.

- **Local Checkers:** We decided to target the internal components (waveform controller, XBAR, int ctrl) of the DUT first and wrote local checkers focused on these components. We developed different sets of local checkers each set targeting one component. Each set had 10-15 checkers based on the component. Getting convergence on these local checkers was very easy as the cone of influence was much smaller for these checkers.
- **End to end Checkers:** We planned for an end to end checker but along with this we also planned for overlapping checkers as we had the opinion of bounded proofs due to design complexity. We had an end to end checker C1 and two overlapping checkers C2 and C3 as shown in Figure 3.

Coverage: With minimum constraints and conclusive checkers, we got good coverage numbers. The only unhit cover points were the expected once due to illegal combination of register values. After waving off these cover points, we got 100% coverage.

IV. RESULTS

The techniques as explained in this paper proved to be very helpful and we closed the verification in less than the expected time. The verification time expected using FPV was 50% less than the time expected using simulation-based verification. We were able to get 100% coverage without any issue. And the full convergence of the overlapping checkers proved to be stronger than end to end checkers. The number of bugs was high as can be seen in the Table 1. Finally, a bug free design was signed off using advance FPV (Traditional FPV + New Techniques). The advance FPV approach helped a lot to left shift the overall verification. The benefits of using the advance techniques are summarized in the table 2.

Table 1. Design and Verification Details

DUT Configuration/ Complexity	Simulation Verification Effort (Proposed)	Formal Verification Effort (Actual)	# Bugs Found	Comments
Number of possible configurations = 2^{2500}	20 Days	7 Days	20	**Formal:** 100% Coverage Less Time **Simulation:** No guarantee for 100% coverage and more time

Table 2. Benefits of advance FPV

Parameter	Impact of advance FPV
Checker Writing Effort	Reduced by 20%
Checker Convergence Effort	Reduced by 30%
Signoff Quality	Improved due to full convergence

V. CONCLUSION

This paper introduces advance FPV techniques which we have used to exhaustively verify a complex design in limited time. The main idea of advance FPV techniques is to divide and concur the DUT which provides better coverage and converged checkers. The key takeaways of this paper are

- Best possible usage of overlapping checkers.
- Technique of one cycle design configuration.

These techniques are proven useful in left shifting the overall verification with 100% coverage.

REFERENCES

[1] Sumit K. Kulshreshtha, Raghavendra J.N., "A Novel Approach to Verify CNN Based Image Processing Unit," DVCON US, 2021

[2] Theo A. Drane, George A. Constantinides, "Leap in the Formal Verification of Datapath," DAC47, 2010

[3] Priyank Kalla, "Formal verification of arithmetic datapaths using algebraic geometry and symbolic computation," Formal Methods in Computer-Aided Design (FMCAD) 2015

[4] Daniel Kroening, Sanjit A. Seshia, "Formal Verification at Higher Levels of Abstraction," IEEE/ACM International Conference on Computer-Aided Design, 2007