2014 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Design Methodology for Highly Cycle Accurate SystemC Models with Better Performance

Simranjit Singh, Infineon, Bangalore, India (Simranjit.Singh@infineon.com)

Prasanth Sasidharan, Infineon, Bangalore, India (Prasanth.Sasidharan@infineon.com)

Sameer Deshpande, Infineon, Bangalore, India (Sameer.Deshpande@infineon.com)

Sandeep Puttappa, Infineon, Bangalore, India (Sandeep.Puttappa@infineon.com)

***Abstract*— with the increasing interest in the architecture exploration and performance analysis, there is an increase in demand of cycle-accurate SystemC models along with simulation performance comparable to that of loosely timed models. With the conventional method of using toggling clock to model timing behaviour, the cycle-accuracy can be achieved but it is not feasible to achieve the desired simulation performance. The other approach is to use clock period to model timing behaviour. It is based on the usage of clock period information for predicting the required clock edges rather than the traditional clock toggling approach as followed in RTL designs. All the processes in the design predict when are to be triggered again based on the clock time period information and schedule the triggering accordingly in the form of event notification and wait statements. The advantage of this methodology is that it is easier to achieve better simulation performance for the software models. However, the models are not always accurate to cycle level and are not synchronized with changing clock. It is difficult to synchronize all the processes in the design when the period of the clock changes. This introduces cycle inaccuracy in the design. With timing behaviour implemented in each process, it is very difficult to debug and to adapt for cycle-accuracy. This becomes more difficult with complex designs. The simulation performance also degrades as the number of processes increase in a design due to increased context switching and the advantage of this approach is lost. There is a need to have a design methodology in place which handles the clock synchronizations and which caters to the timing requirements of all the processes in a module. This paper describes a design methodology which refines this approach to achieve the desired cycle-accuracy and the simulation performance. It has already been proven on multiple designs**

Keywords—Simulation, Performance, Cycle Accuracy, Fast models, Re-usability, RTL, Extensibility.

I. INTRODUCTION

Virtual prototyping has become essential to handle the increasing design complexity and reducing time-to-market windows. The virtual prototypes are being used for various purposes like early software development, software performance analysis, architectural exploration and hardware performance analysis. The level of abstraction of functional and communication aspect of a model largely depends on the use case.

There is always a trade-off between accuracy and speed of a model. For software development and software performance analysis, a less-accurate but highly fast model is required. For architectural exploration and hardware performance analysis, a highly accurate, both functionally and temporally, and fairly fast model is required. As the industry's interest architectural exploration and hardware performance analysis is increasing, the demand of cycle-accurate models with performance equivalent to that of loosely-timed model is increasing. It requires change in the modeling techniques to achieve cycle-accuracy with the desired performance. The paper describes one such methodology to model the desired virtual prototypes.

II. TRADITIONAL APPROACH OF MODELING FOR CYCLE-ACCURACY

The traditional approach to model for cycle-accuracy is to bring the virtual prototype or model to the hardware. The model is designed to work at each clock edge where the clock is a toggling signal. It involves a large number of processes executing at each triggering edge of the clock which causes simulation speed to fall drastically.

The other approach is to use clock period to model timing behaviour as shown in Figure 1. This approach is based on the usage of clock time period for predictions of required edges rather than the traditional toggling clock approach, which results in a significant increase in simulation performance. The timing behaviour is modelled

through each process scheduling itself at the required point of time in simulation using timed event notifications and wait statements. For example, if a process is triggered at ever second rising edge of the clock, then it is required to be re-scheduled to run after two into the clock period.

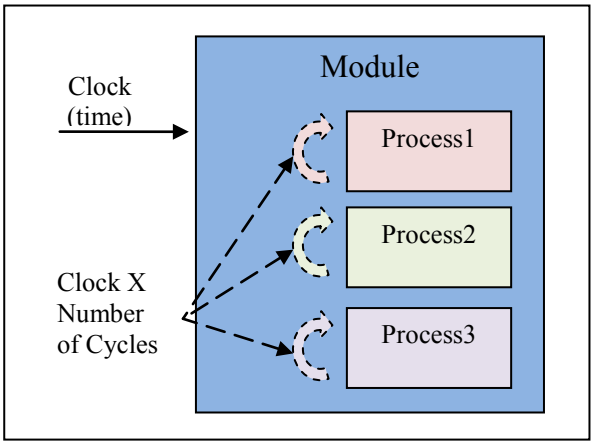

Figure 1: Model designed with traditional approach

However, it does not assure cycle-accuracy in all the cases. In case there is a change in the clock period, the cycle-accuracy will be lost as each process is scheduled using the old clock-period information and thus the operations are not carried out in synchronization with the clock edges of the new clock. This introduces cycle inaccuracy in the design. With timing behaviour implemented in each process, it is very difficult to debug and to adapt for cycle-accuracy. This becomes more difficult with complex designs. The simulation performance degrades as the number of processes increase in a design due to increased context switching and the advantage of this approach is lost.

III. CLOCK CONTROLLER APPROACH OF MODELLING FOR CYCLE-ACCURACY

This article describes a design methodology to refine the clock period based approach. The idea is to have a central unit that handles scheduling of all the processes in the design and takes care of changes in the clock period. The methodology provides a generic clock control unit which acts as a single source of clock-information in a design. It recommends a design to be modularized where the sub-modules implement clock-dependent operations in call-backs registered with clock control unit. The clock control unit maintains the operations synchronized and handles changes in the clock-period. It assures cycle-accuracy in all cases. With this new flow, it would be easy to model any IP systematically thereby leading to a maintainable design which is extensible for future design needs.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

A. Flow

The new design method demands the design of an IP to be modularized. An IP is functionally divided into smaller sub-modules. The sub-modules are categorized on the basis of functionality and the associated clock domain. The clock control unit also becomes part of the design as a sub-module as shown in Figure 2. It is connected to the module's input clock and to all other sub-modules which require clock to operate via registered call-backs.

In the context of clock control unit, the registered sub-modules are referred as clock-clients. As in the traditional approach, a module would implement all the clock dependent operations in form of processes, sensitive to the module clock. Each process would be responsible for re-scheduling itself by calculating the next trigger time using next required clock edge. With clock control unit, in the sub-modules or clock-clients, the required clock-dependent operations are modelled in the clock call-back registered with the clock control unit, instead of processes. As the sub-modules are independent of the timing information they do not use processes or any other SystemC constructs which assist in scheduling. Ideally, there shall not be any process in a clock-client.

Hence, the sub-modules become more C++ style classes rather than typical SystemC modules. It reduces the overhead associated with the processes in the design and context switching during the simulation. It results in significant increase in the simulation speed.

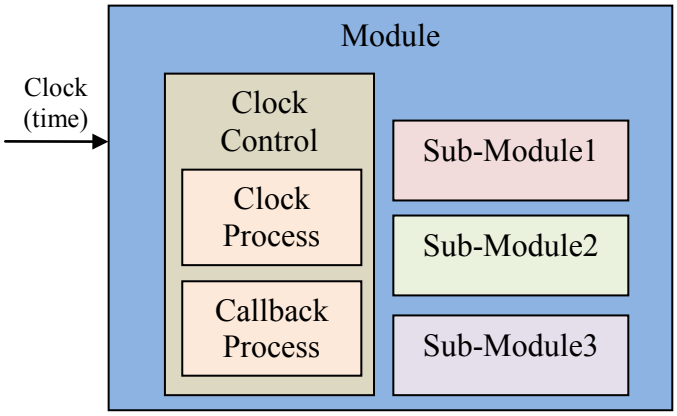

Figure 2: Model designed with new approach

This approach also makes it easier for the designer to track the flow of the design and provides better debugging possibility.

IV. METHODOLOGY COMPONENTS

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the "Save As" command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

A. Clock Control

Clock-Control is a generic block which can be used in any module implementation. It runs on the module's clock and effectively handles the clock-based scheduling of module's processes. As discussed above, it is used as a sub-module in a design and all the other clock-driven sub-modules are registered with it, known as clock-clients. Figure 3 shows the overview of clock controller & its clients.

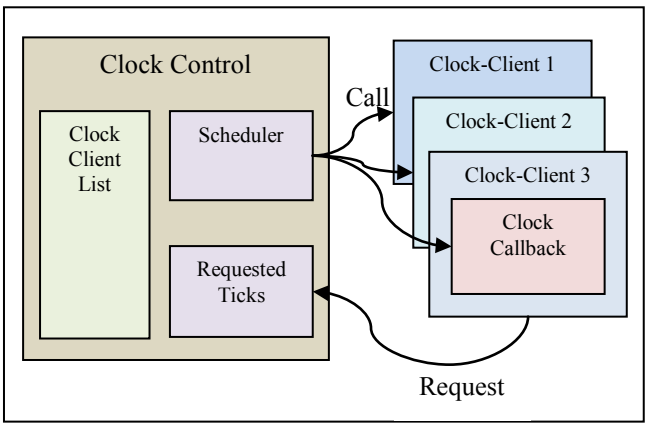

Figure 3: Call-request mechanisms in Clock Control

The clock control unit works on the request-call mechanism where the registered clock-clients request for a call after the desired number of clock-cycles and clock control invokes registered call-backs on the clock-clients. The number of cycles requested by a clock-client is also referred as clock ticks. The clock-control keeps track of the number of cycles, or ticks, requested by clock-clients and chooses the minimum of the requested ticks. After the elapse of selected ticks, it calls the corresponding call-back for all the clock-clients. Figure 4 explains the flow of operation of the clock controller along with its registered clock clients.

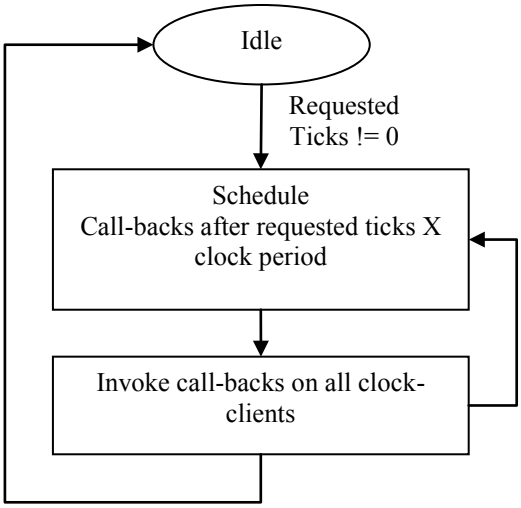

Figure 4: Clock Control call-back and request flow

In the call-back, a clock-client shall check whether the clock tick count provided with the call-back matches the requested count or not. That is, if the clock tick count at the time of clock tick request was X, then the current clock tick count shall be X + requested ticks. If it is a match then the clock-client shall perform the needed operation. If it is not a match, the clock-client shall re-evaluate the needed ticks and register it with the clock-control.

The clock control library provides a base class for the clock-clients which takes care of the operation of comparing ticks with the clock tick count before the call-back of the derived clock-client (sub-module of the design) is invoked. If the check fails, the call-back on the sub-module will not be called. It is as shown in the following Figure 5.

A clock-client can register for needed clock ticks for the next round during this call-back. This method of registering clock-ticks is referred as synchronous registration. If there is no such request from any clock-client, the clock-control will be idle and will not trigger any call-back.

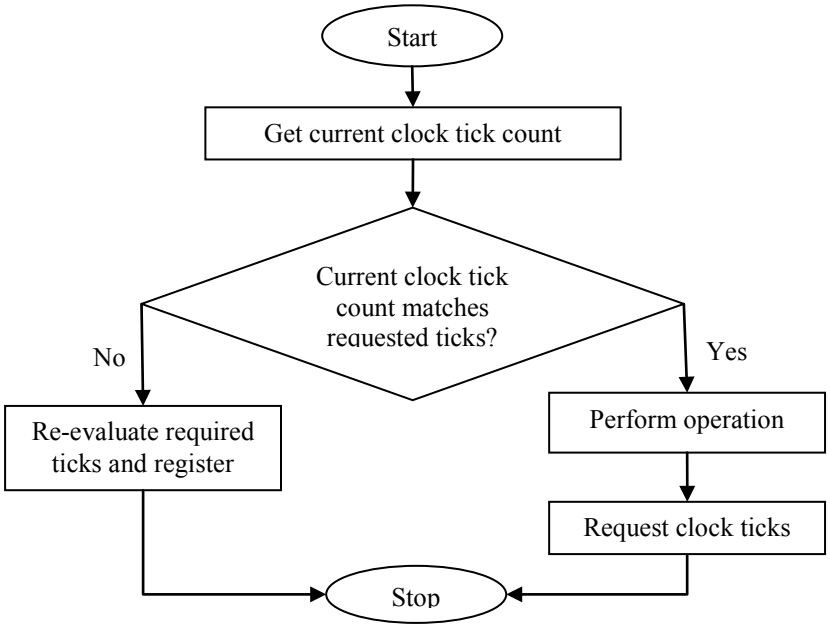

Figure 5: Base clock-client call-back

When in idle mode, the clock-control can be activated by a clock-client by registering clock ticks asynchronously.

B. Clock Triggering Edge

A design can work on positive edge or negative edge or, in some cases, on the both the clock edges. The clock-control block can provides call-backs on both, positive and negative, edges of the clock. A clock-client can register the triggering-edge it needs to work on with the clock-control and clock-control shall provide the corresponding call-back to the clock-client.

1) POSEDGE

a) It indicates that the clock-client works only on the positive edges of clock. All the ticks requested by such a block will be considered as positive edge ticks.

2) NEGEDGE

a) It indicates that the clock-client works only on the negative edges of clock. All the ticks requested by such a block will be considered as neg-edge ticks.

3) ANYEDGE

a) It indicates that the clock-client works on both, positive and negative, edges of the clock. All the ticks requested by such a block will be considered as sum of positive-edge and negative-edge ticks.

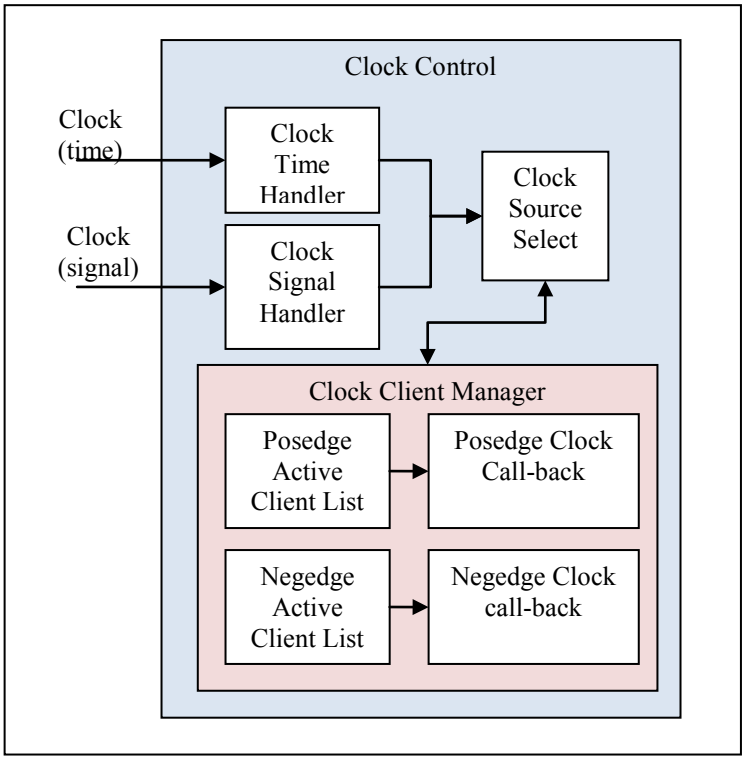

Figure 6: Structure of clock control unit

The clock-control works with triggering-edge configuration as set by the clock-clients. In a design which has clock-clients working on different edges, the clock-control will always work in the ANYEDGE configuration. It shall take care of calling the correct call-back for a clock-client according to the triggering edge registered for a clock-client.

C. Clock Source

In SystemC design approach, the timing behaviour is modelled using period information of the clock. But in case the module is configured to run on external clock which is generally a signal or a toggling clock, determining period can lead to temporal in-accuracies. Clock source select block selects the desired clock input for the clock control to work on.

The clock-control provides the facility of using both, the period based clock and clock signal, as clock source. It also supports switching between clock sources on the run. A clock-client responsible for selecting clock shall configure clock-control to use the selected clock source.

1) Period Based Clock

a) The clock-control uses the period information to trigger call-backs for the clock-clients. It calculates the pos-edge or neg-edge of the clock using period and the simulation time to get the number of edges passed. It uses this information to schedule triggering of call-back after the requested ticks.

2) Signal Based Clock

a) The clock-control can also work with a signal type clock where it is like a real toggling clock. In this configuration, it keeps track of the number of edges passed and triggers call-back when it reaches the requested number of ticks (edges).

D. Clock Frequency Changes

In the traditional approach discussed above, the main challenge is to handle the clock frequency or period changes in the module. As all the processes have to be re-scheduled and perform duty cycle adjustments, it becomes un-manageable. In the clock control unit, the clock changes are handled centrally. In case there is a change in the clock period, the call-back is re-scheduled according to the new clock period. It also performs the duty cycle adjustments which become critical in case the clock is updated at time other than clock's rising or falling edge.

E. Clock Clients

As per the definition, a clock-client is a module which is derived from the "ClockClient" class from the clock control library. The base class performs the checks on the clock tick count and requested ticks before calling the call-back on the derived clock-client. It handles re-evaluation of the requested ticks if required. It also performs the requested ticks' adjustment depending upon the triggering edge configuration of the clock-client and the clock control.

In a design, a clock-client can be active or reactive. An active clock-client is the one which puts the requests to the clock control for the call-back, for example, a clock-client, input sampler, that needs to sample external input on every edge of the clock. A reactive clock-client is the one which does not put any request to the clock control but performs the operations when the clock call-back is invoked. Figure 7 explains the active and reactive clock clients. For example, a clock client that works on the output of input sampler.

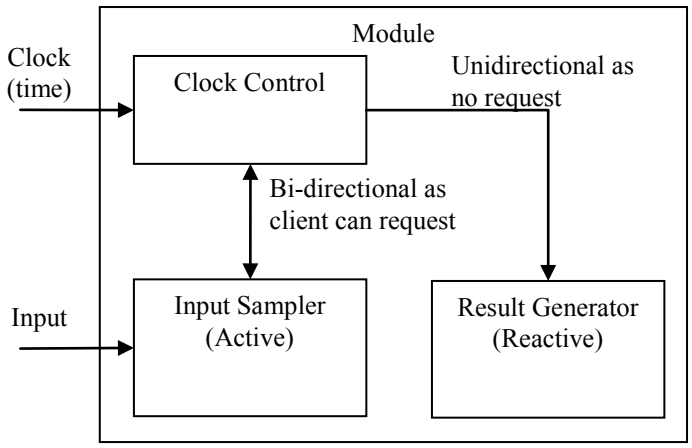

Figure 7: Active and reactive clock clients

In general, a design will have one or two active clock-clients which request for the clock ticks whereas others will be reactive. It makes the design simpler as the clock tick count checks in the base clock-client will be performed for one or two clock-clients rather than all the clock-clients in the design.

V. BENEFITS OF CLOCK CONTROLLER METHODOLOGY

- Centralized clock controller sub-module that takes care of clocking for the module so that all the other sub-modules are independent of the timing information. Clock control is a reusable sub-module which can directly be used in other modules.
- This new approach minimizes the use of SystemC processes and events to reduce context switching and hence increases the overall simulation performance.
- Provides the flexibility to adapt to changes by providing plug and play kind of feature for different functionalities inside a module there by making the design extensible. If any new feature is to incorporated in the module then the additional functionality can be added by introducing a new sub-module.
- Provides the flexibility in design to be adapted for cycle accuracy related changes in all the cases.
- Gives the designer more control over the clock synchronization and events.

VI. APPLICATION EXAMPLE AND RESULTS

This methodology was applied to a module called DSADC (Delta-Sigma Analog to Digital Converter). This module was developed with the older approach, has a single SystemC module which uses much number of processes (more than ten) and events to model the needed functionality. The timing behavior is modelled using timed event notifications and wait statements to trigger processes which made the implementation very complex. It also turned out to a difficult job to debug and trace the flow of the functionality of the module.

With the new methodology, the DSADC module is divided into various sub-modules namely filters, integrator and modulator sub-modules and the timing behavior is modelled using the Clock Control sub-module. With the clock control block taking care of the timing requirements of the module, it reduced the burden on the other sub-modules and the timing concerns were handled centrally by the clock control. The number of processes in the design were reduced from thirteen to five (including clock control).

To test the improvements on simulation performance of the new design method on the DSADC, a set of stress tests which already existed for the old module, were run with the newer module. The average time taken for a particular stress test was calculated considering several runs with both the old and new module. The approximate figures of the wall clock time measured with the older module was around 131878 milliseconds and whereas with the new module for the same test the average figure is around 29990 milliseconds. With the newer design approach the performance increase turned out to 77% faster compared to the older module for the same stress tests.

There is a significant improvement in the performance with the new design approach. The design is also now easy to maintain and adapt for the cycle-accuracy changes. The module was also tested with the RTL verification environment which established the functional and cycle accuracy of the design.

VII. CONCLUSION

With the clock control, it is easier to create models that are cycle-accurate and have fast simulation speed. The clock control unit is a generic block and can be used in any design. The modularized designs are easy to implement as the timing information and scheduling is handled by the clock control unit. By minimizing the number of processes in the design, the simulation speed can be increased significantly. It is easier to debug and adapt for the changes. The improvements become more significant with complex designs. The approach makes the designs more flexible, extensible and easy to debug.