SoC Firmware Debugging Tracer in Emulation Platform

Kubendra Kumbar [1]
SSIR- MST
Samsung Semiconductor India
Bangalore India
kubendra.sdp@samsung.com

Sandeep Vallabhaneni 2
Controller Development Team
Samsung Semiconductor
Suwon South Korea
v.sandeep@samsung.com

Ken Joseph Kannampuzha[3]
SSIR –MST
Samsung Semiconductor India
Bangalore India
ken.joseph@samsung.com

Hojun Shim [4]
Controller Development Team
Samsung Semiconductor
Suwon South Korea
hojun.shim@samsung.com

Byung Chul Yoo [5]
Controller Development Team
Samsung Semiconductor
Suwon South Korea
byung.yoo@samsung.com

***Abstract*- Firmware debugging is one of the biggest challenges in System on Chip (SoC) designs. Currently co-verification can be done with lot of solutions are not compatible, infeasible, expansive and takes more gates counts with SoC platform prototypes , unavailability of required system design debugging due to limitation of breakpoint feature like with In-ciruit Emulation(ICE) or software debuggers and also not capture the exact dynamics of the system due to its limitation ,hence it is difficult to reproduce the exact bugs or fixing the issue .**

An optimal and efficient SoC firmware Performance/debugging analysis mechanism using firmware tracer fine-tuned to emulation platform.

Key Words: SoC, Cortex M3, Cortex M7, Firmware debugging, Debugger Tracer, product firmware, Emulation, Pallaindium , software debugging ,Palladium XP Verification ,Co-verification,

I. INTRODUCTION

Co-verification is biggest challenges in multi-processor system on chip (MP-SoC) designs and in that again challenging is debugging the firmware issues in the hardware platform .In our current flash SoC controller design shown in the Figure 1 , in this we are not able to add the Advance Risc Machines (ARM) based embedded Trace module (ETM) for the cortex based M3& M7 processors debugging in chip design & also in PXP emulation platform due to chip area , cost considerations and resource crunch in the emulators . To overcome the above problem and alternative debugging trace solution is proposed with less gate count and cost.

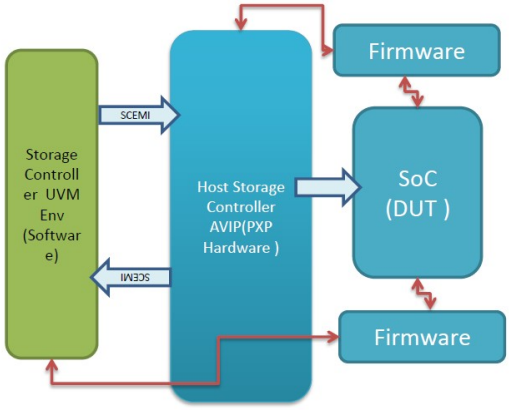

Figure 1: Flash SoC controller in PXP Emulator.

This firmware debug tracer will debug the software/firmware component that enables the reconstruction of program execution of each firmware functions on the simulation waveform. Helps in optimizing the firmware components & also used as feedback for the next chip tape outs /revisions for the better performance by changing in the architectural considerations with respective hardware and software segregations.

The way how waveforms debugging will help in the hardware modules, same manner firmware debugging is also done in cadence simvision with help of this proposed solution. This is implemented by current firmware function that is being executed and the hierarchical level of the Firmware functions under execution plot with help of corresponding Program Counter (PC) value and searching this function name based on the function address.

In this paper, we are proposing feasible solutions to develop the firmware (FW) tracer for the Cortex M3/M7 series processors and it helps for he where it get struck and also where we can optimize the firmware functions by plotting in waveforms of the function name.

II. FIRMWARE TRACER

To implement the Firmware Tracer, we need to know the information about the function names for every corresponding Program Counter value with respective cortex M3/M7 processors and also to predict the correct program counter value depending on the PUSH/POP conditions considering the Branch and Branch booster conditions, ISR interrupts and Branch with link instruction etc.

This function name and function address is extracted from the given firmware code using Perl scripts and system level addressing. For this searching logic, we can use a for-loop to search the function name based on the function address and requires a huge number of resources in Hardware (approximately 800K logic gates for 900 functions). Therefore, to save logic gates, we need to implement the function name searching logic in Software and have a mechanism to communicate between Software and Hardware to get the function name based on the Program counter value. The SCEMI-pipe based implementation serves good for this purpose.

Please find the below Figure 1 for the block diagram for the Firmware Tracer architecture using Standard Co-Emulation Modelling Interface (SCEMI) pipes. It having the Hardware and software parts & communication between these two is done with SCEMI pipes.

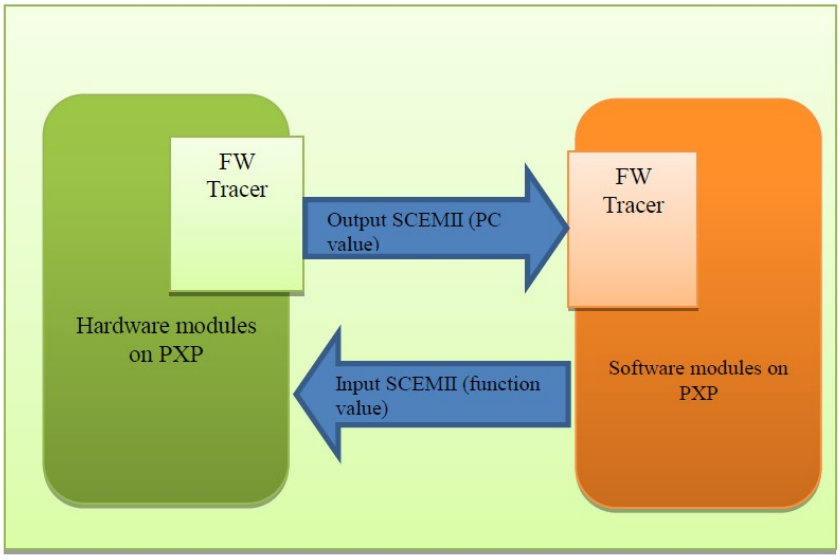

Figure 2 : Firmware Tracer architecture in PXP emulation

This FW tracer makes use of SCEMI-pipes to send the program counter value to the Software part and get back the function name based on program counter value. This communication must be allowed to happen only during waveform dumping; otherwise, this SW to HW communication becomes an overhead and degrades the performance.

III. IMPLEMENTATION

Firmware Tracer can be implemented using the hardware resources only if there is no limitation for the hardware resources requirement. Currently for the 900 function name (40 characters of function name) support implementation, design it is taking around taking around 800K gates. To save the hardware resources with optimal simulation speed, combination of both hardware and software logic using the SCEMI pipes / DPI (Direct Programming interface) based MARG(Cadence specific interface) is good choice . Here pseudo code using SCEMI pipes are explained.

A generic data file (DAT) is created for the given unit level/product level firmware with respect to function names & function address using the script language. After the loading the real firmware code in terms of ITCM (Instruction tightly coupled memory), DTCM (Data Tightly coupled-Memory) and ROM code into the test bench.

Next challenge is to find the current program counter (PC) value for the cortex processors and searching logic for the function names depending on the current PC value. Here we are using cortex (M7) processors for the SoC implementation and predicting the PC value for the PUSH & POP operations is challenging due to

- Branch and Branch booster conditions,
- Interrupt service Routine (ISR): Interrupts entry and exit
- and Branch with link instruction
- complex and multi stage pipeline architectures

Below are the steps for implementing the SCEMI based FW tracer:

1. Here we use the SCEMI-pipes to send the program counter value to the Software(S/W) part and get back the function name based on program counter value

2. Define SCEMI output pipe/MARG interface on HDL-side to send the Function_Label_Addr[31:0] and PC value from HW side to SW side. Element size of SCEMI output pipe is 4 bytes (32-bit Program Counter).

```
SCEMI_INPUT_PIPE #(.Bytes_Per_Element (1),Payload_Max_Elements(1),Buffer_Max_elements(32)
my_output_pipe()) ;
    // parameter BYTES_PER_ELEMENT = 1;
    // parameter PAYLOAD_MAX_ELEMENTS = 1;
    // parameter BUFFER_MAX_ELEMENTS = 32;
    // localparam PAYLOAD_MAX_BITS = PAYLOAD_MAX_ELEMENTS * BYTES_PER_ELEMENT * 8;
```

Using the blocking send method, PC value to software module for the function name searching.

```
my_output_pipe.send(1, PC,1 );
// task send(
// input int num_elements, // input: #elements to be written
// input bit [PAYLOAD_MAX_BITS-1:0] data, // input: data
// input bit eom, // input: end-of-message marker flag
  <implementation goes here>
// endtask
```

3. Define SCEMI input pipe on HDL-side to receive the Function_Label_Name[319:0] from SW side to HW side. Element size of SCEMI input pipe is 40 bytes (40 characters of function name).

SCEMI_INPUT_PIPE #(.Bytes_Per_Element (1),Payload_Max_Elements(1),Buffer_Max_elements(256) my_input_pipe()) ;

// parameter BYTES_PER_ELEMENT = 1;
// parameter PAYLOAD_MAX_ELEMENTS = 1;
// parameter BUFFER_MAX_ELEMENTS = <implementation specified>;
*//localparam PAYLOAD_MAX_BITS = PAYLOAD_MAX_ELEMENTS * BYTES_PER_ELEMENT * 8;*

Using the blocking receive method, function name is received from the S/W module for the plotting waveform

my_input_pipe.receive (1, num_valid,function_name,end_of_message);

task receive(
// input int num_elements, // input: #elements to be read
// output int num_elements_valid, // output: #elements that are valid
// output bit [PAYLOAD_MAX_BITS-1:0] data, // output: data
// output bit eom, // output: end-of-message marker flag
<implementation goes here>
// Endtask

The size of the function name is changed depending upon the requirement of projects A switch is used to have configurable size for the function name.

4. Create an System Verilog (SV) file with sv input and output scemi pipes and connect these pipes to the SCEMI pipes on the HDL side.
5. Add logic in SV file to get the function address from HDL side using SCEMI output pipe, find the corresponding function name and send it to the HDL side using SCEMI input pipe.

// Hardware Part: Glue logic for the finding the PC value depending upon various PUSH/POP conditions using
the cortex M7 signals like bl_ext2 , br_ex2 , int_exit , int_entry ,pc_ret,_rege ,pop_to_pc_iss and br_pc_ex2 etc .

// Software Part: searching logic (using the for loop) for the function names depending upon the provided PC value

6. Call the send and receive tasks of SCEMI pipes on HDL side FW tracer during waveform dump.

Figure 3 shows the generic example of how FW tracing done, How the each functions of firmware will be displayed in the waveform with help of PC value of the processors , Please find the successful SoC tape out simulation results for the PUSH conditions(Figure 4) , POP conditions (Figure 5) and both PUSH & POP conditions (Figure 6) .

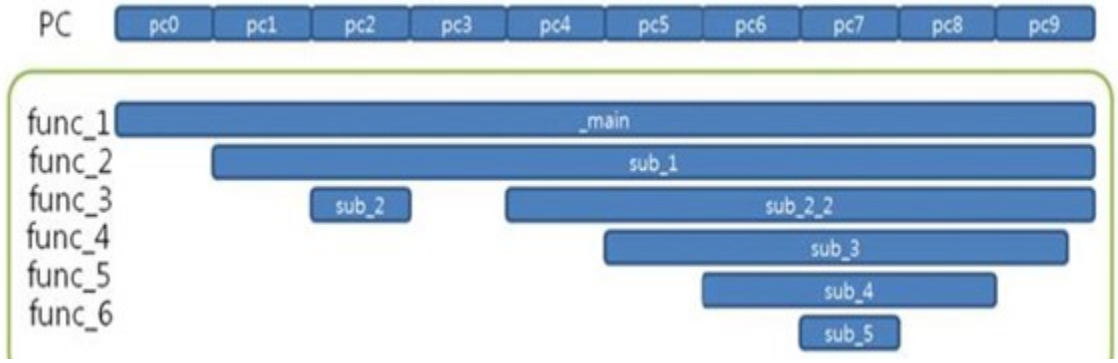

Figure 3-Firmware Tracer operation

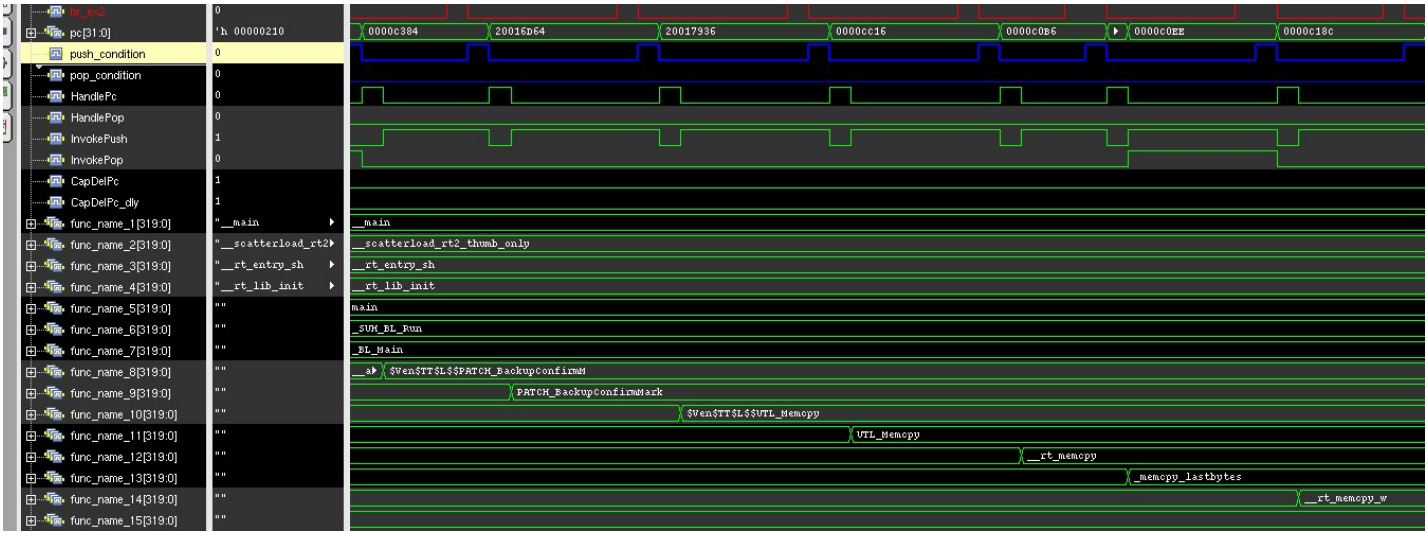

Figure 4: Simulation results for the PUSH Conditions with function name & PC value

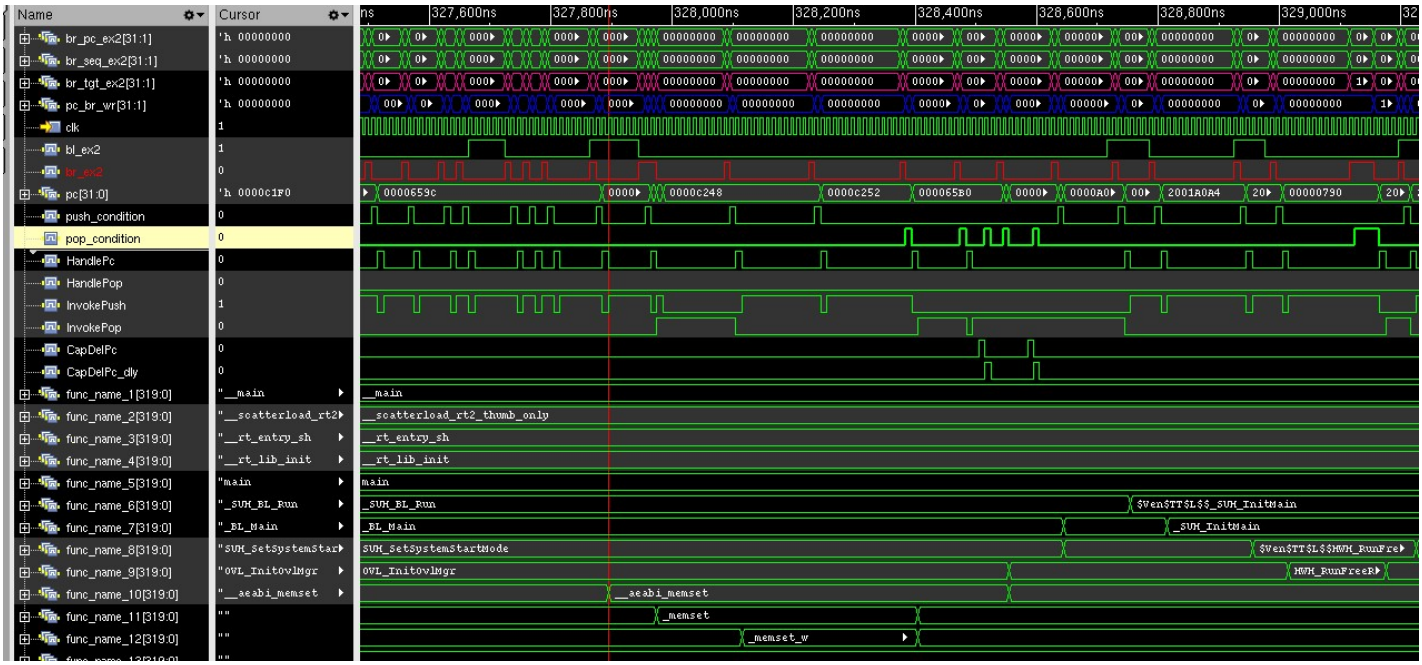

Figure 5: Simulation results for the POP Conditions with function name & PC value

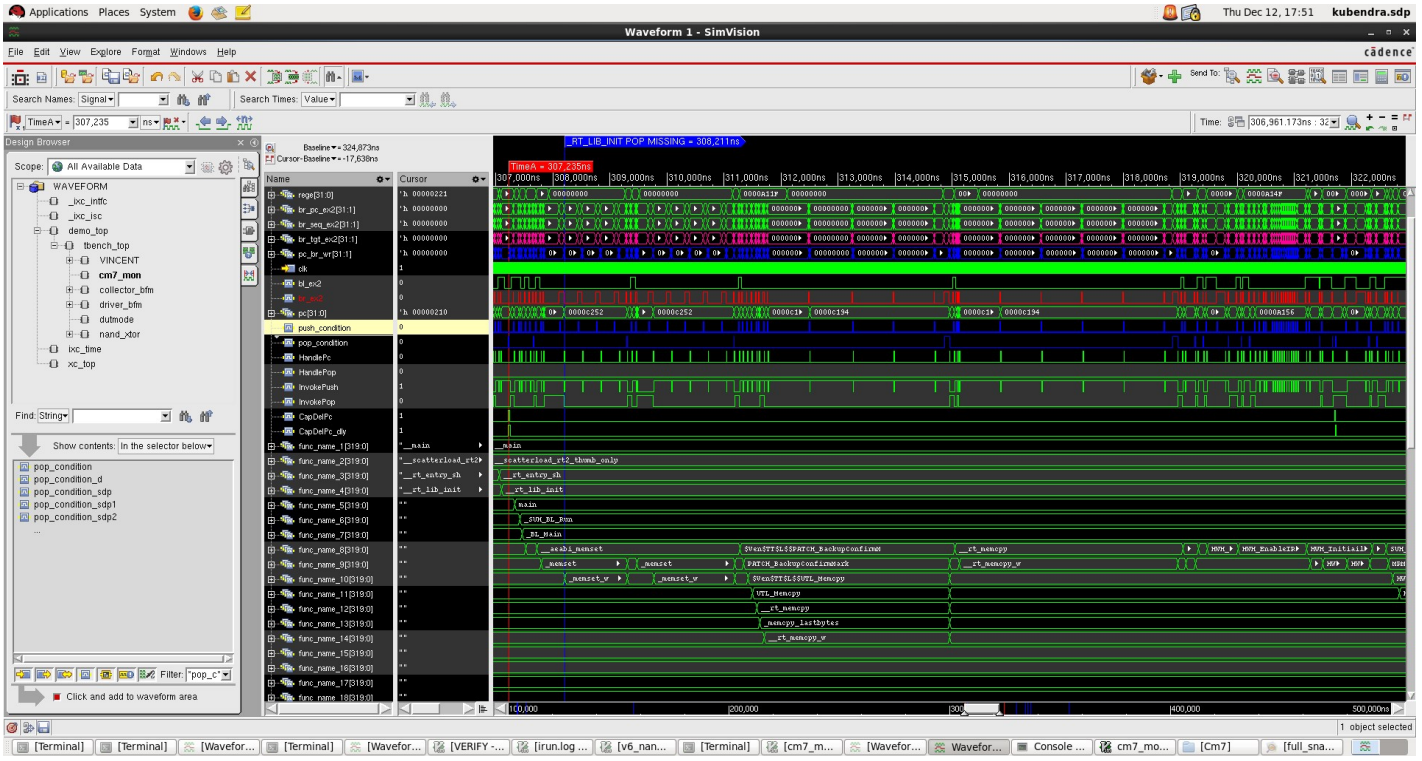

Figure 6: Simulation results for the PUSH & POP Conditions with function name & PC value

IV. CONCLUSIONS

The PC value predicted will vary from processor to processor and depending upon the tradeoff between gate count and speed, emulation hardware and software optimization is done by switching the Hardware/Software logic using the DPI/SCEMI /MARG mechanisms in the Emulation This paper we are achieving the considerable speed by doing "function name searching logic with software part and Program counter value logic with hardware part using the SCEMI pipe based interface.

For the Cortex M3 series, finding the PUSH and POP operations and plotting the Functions with PC value is straightforward. In case of Cortex M7 it is little challenging due to miss predication of the PC value for the PUSH & POP operations, Branch and Branch booster conditions, ISR interrupts and Branch with link instruction. Please find the Figure 6 for the SoC simulations results for the Cortex processor. With these waveform display we can easily debug the firmware code, optimize the software and hardware blocks and can switch logic (from hardware and software and vice versa) for the better performance results.

IV.REFERENCES

[1] Yael Abarbanel , Eli Singerman, and Moshe Y .Vardi , "Validation of SoC firmware-hardware flows : challenges and Solution directions",51st ACM/EDAC/IEEE Design Automation Conference (DAC), 2014.

[2] Cadence Palladium emulation user manual.

[3] S. Gopikrishna ,Manoj Jha,S,Sreekanth,G Savithri , "A multiprocessor System On Chip verification on hardware accelerator and Software Emulation," IEEE International Conference on Advances in Electronica, communications and Computer Technology (ICAECCCT) 2016.

[4] ARM Cortex M3 and M7 user manuals.

[5] ARM ETM user manual.

[6] Accellera Universal Verification Methodology.

[7] Special Thanks to Mr.Boopathy Prabhaharan A form the SSIR Firmware team for the ARM processor architectures and script support.