DVCon India 2023
Abstract submission template

TITLE OF PAPER	
AUTHOR 1	**Name: Akhil Arora** **Organization: QUALCOMM India Private Limited** **Job Title: Engineer, Senior Staff** **Email ID: akhiaror@qti.qualcomm.com** **Mobile no: 9711333139**
AUTHOR 2	**Name: Sonik Sachdeva** **Organization: QUALCOMM India Private Limited** **Job Title: Engineer, Principal/Manager** **Email ID: soniks@qti.qualcomm.com** **Mobile no:**

Title : Wrong clamps can kill your chip!!.... find them early

ABSTRACT

Validating Isolation strategies is one of the most challenging aspect of Power Aware Verification. Static Verification has proven to be a very good methodology to solve the pertinent issue of which signals between the power domain crossings, need to be isolated i.e. where the isolation cells should be placed for the power domain crossings, but this methodology comes out short on verifying the correct clamp values for the isolation strategies. This is where the Dynamic Power aware simulations play a very important complementary role.

However, with the increasing complexity of the today's SoCs, running dynamic simulations to test and verify all the clamp values would require running many test scenarios, requiring a lot of time and effort. It becomes more challenging with the ever reducing, "time to market", which can impact the PA verification timelines, thereby resulting in ECOs, by the time dynamic simulation catches the isolation strategy related bugs.

This paper proposes a methodology to left shift the verification of the specified clamp values, through a guided signoff process.

Introduction

The Clamping Challenge : Only the dynamic simulations can reveal any real issues with the clamp values associated with the isolation strategies in the UPF. With the help of dynamic simulations, the functional tests can verify the correct values of the isolation cell outputs. However, with the growing size of the modern SoCs and the growing complexity of the power architecture, it is becoming increasingly difficult to code all the possible scenarios at the SoC level, to validate all the low power artifacts in the SoC architecture. Especially for the isolation cells, to have the complete coverage for the clamp values, though desirable, but is extremely challenging.

The Power aware dynamic simulations continue to happen till the later stages of the verification due to which any bugs due to the wrongly specified clamp values can come up very late in the verification cycle, resulting in design ECOs.

In the following sections, the paper will describe a signoff based approach which can be initiated very early in the PA verification schedule to catch the wrongly specified clamp values. This process can help catch the isolation bugs very early in the cycle thereby increasing the Quality of verification and also since the approach suggested by this paper is signoff based so by catching the bugs early it helps in saving the debugging effort as compared to, if the bugs were to be uncovered through the dynamic simulations later in the cycle.

Proposed Solutions

Isolation cell outputs can drive different type of values, i.e. 0,1,Z,Latch - based on the functional requirement. Values of type Z,latch have limited applications; more prevalent type of isolation output values are the constants, 0 and 1. Generally these values are decided based on, how the receiving power domain can be kept isolated from the corruption in the driving power domain. Isolating receiving logic would mean that the isolation values should not trigger any functionality i.e. the receiving logic should be dormant or in the reset state. That is why mostly, the 0,1 isolation output values are decided based on the active reset values of the signals which are being isolated by the specified strategies.

Thus, generally, most of the isolation strategies would be driving the reset values onto the isolated signals, except for Z or latch type isolation strategies or for some very specific application dependent strategies.

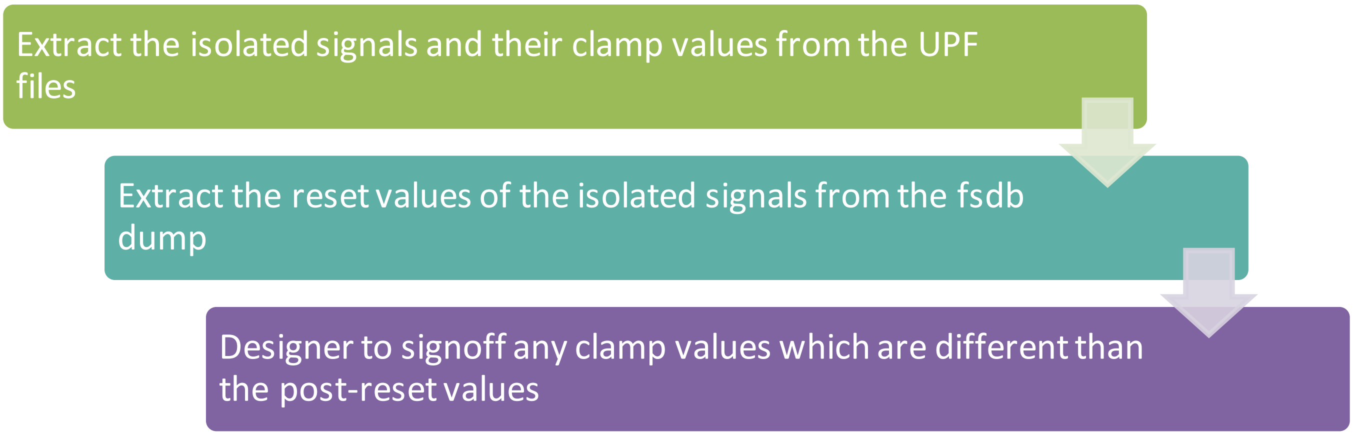
A quick, comparison, between the reset values of the isolated signals with the clamp values specified in the isolation strategies can provide the list of the isolated signals which are assigned clamp values, different from the reset values.

Since, list of isolated signals with non-reset clamp values will have only limited number of unique strategies, so this list can be easily reviewed with the designer to find out if these are the correct isolation values or have been wrongly assigned. The designer can also catch isolation strategies which wrongly specify the reset value for the clamp values for which non-reset value is expected.

Similarly the Z or latch type isolations can also be reviewed separately to make sure that those are also correctly assigned as per the design requirements.

This sign-off process will, not only catch the bugs related to the wrongly specified clamp values in the isolation strategies but will also provide a significant left shift to the overall power aware verification strategy as this sign off can be done very early in the cycle. It will also reduce the overall debugging efforts as compared to, if the isolation related bugs were to be uncovered in the functional simulations, as tracing back to the source of the incorrect clamp value would be a significant effort for the verification engineer. The early coverage provided by this methodology will also strengthen the confidence in the PA verification.

Clamp Value Vs Reset value comparison flow


Extract the isolated signals and their clamp values from the UPF files
Extract the reset values of the isolated signals from the fsdb dump
Designer to signoff any clamp values which are different than the post-reset values

1. Extract all the isolated signals and the clamp values from the UPF files
    a. Searching all the UPF files used at the SoC level may not be straight forward due to the modular nature of specification and how the files may be distributed across various locations
    b. VCS dumps various reports related to the power aware artifacts while doing the elaboration, one such report has all the information about the inserted isolation cells
        i. <compile_area>/mvsim_native_reports/isolation_insertion_info.txt : This report provides information regarding all the isolated signals which are inferred from the isolation strategies mentioned in the UPF files
            1. Report format : <isolation_control> <isolation_sense> <clamp_value> <isolated_signal>
        ii. All the isolated signals and their clamp values can be extracted from this report for further comparison

2. Extract the reset values of the isolated signals from the fsdb dump
    a. For the isolated signals which are extracted in the step-1, the reset values can be extracted from the fsdb dump of one of the sanity test
    b. VCS tool provides multiple mechanisms to extract this information from the available fsdb database
        i. Fsdbreport utility to extract the values of the isolated signals at the reset event

Sample fsdbreport command:

```
Unix>> fsdbreport <fsdb_name> -s <list of all the isolated signals> -strobe <reset_event>
```

ii. C-APIs to access FSDB information through the VCS Native Programming Interface

<u>SAMPLE NPI code to extract values from FSDB</u>

```
FILE * sigFileH;
 sigFileH = fopen ("isolation_sig_list_without_latch_type", "r");
 if (sigFileH == NULL) {
     printf (" Could not open the signals file list sig_list");
     exit(1);
 }
 while ((read = getline(&line, &len, sigFileH)) != -1) {
 line[read-1]= '\0' ;
 npiFsdbSigHandle sig = npi_fsdb_sig_by_name( fp, line, NULL);
 if ( sig ) {
     // create vc traverse handle
     npiFsdbVctHandle vch = npi_fsdb_create_vct( sig );
     if ( vch ) {
         // go to max time for initialization
         npi_fsdb_goto_time( vch, resTime);   // Move to the time of reset
         // get the time & value information of current vct handle
         npi_fsdb_vct_time( vch, &currTime );
         npi_fsdb_vct_value( vch, &val );
         printf("%s Time = %10llu, value = %s\n", line, currTime, val.value.str );
                if (strcmp(val.value.str ,"1") == 0) {
                          printf ("high %s\n",line);
                }
                else if (strcmp(val.value.str ,"0") == 0) {
                          printf ("low %s\n",line);
                }
                else if (strcmp(val.value.str ,"x") == 0 ) {
                          printf ("Unknown %s\n",line);
                }
                else if (strcmp(val.value.str ,"z") == 0) {
                          printf ("ZZZZ %s\n",line);
                }
     }
 else
     printf("Failed to create vch for sig: %s.\n", line);
 }
 else {
     printf("Failed to find sig: %s.\n", line);
 }
 }
 fclose(sigFileH);
```

c. ***At SoC level, where the size of the FSDB database is huge and the number of the isolated signals is very large, the C-APIs are much faster, efficient, and scalable, in extracting the information from the FSDB database.***

Results

Using the mvsim_native_reports/isolation_insertion_info.txt report and the reset values extracted from the fsdb database, a report in the below format was created for the designer review.

signal name : <hierarchical name of the isolated signal> , iso_spec : <low/high> , reset_value : <low/high>

This report is very easy for the designer to review and provide a signoff.
Using this methodology, we have been able to catch several wrong clamp value related bugs very early in the verification cycle for the recently executed projects.

Conclusions

SoC power aware verification is very challenging and involves a lot of effort. Validating the clamp values for the isolation strategies is one of the most important aspect of PA Verification. Wrong clamp value related bugs are not easy to detect and it is extremely challenging to get 100% coverage using the fucntional tests.

The proposed solution provides a sign-off process for the verification and design team to achieve early verification of the clamp values specified in the UPF files to ensure that the related bugs can be caught early in the cycle, thereby saving the time and effort during the power aware verification.

References

[1] VC Apps Native Programming Interface (NPI) Documentation : Reference for C-APIs