SAND2016-1856C

Extending the Constrained Random Simulation Methodology into Physical Device Verification of an Embedded Processor ASIC

Anita Schreiber

High Integrity Software Systems, Department 2622

Sandia National Laboratories

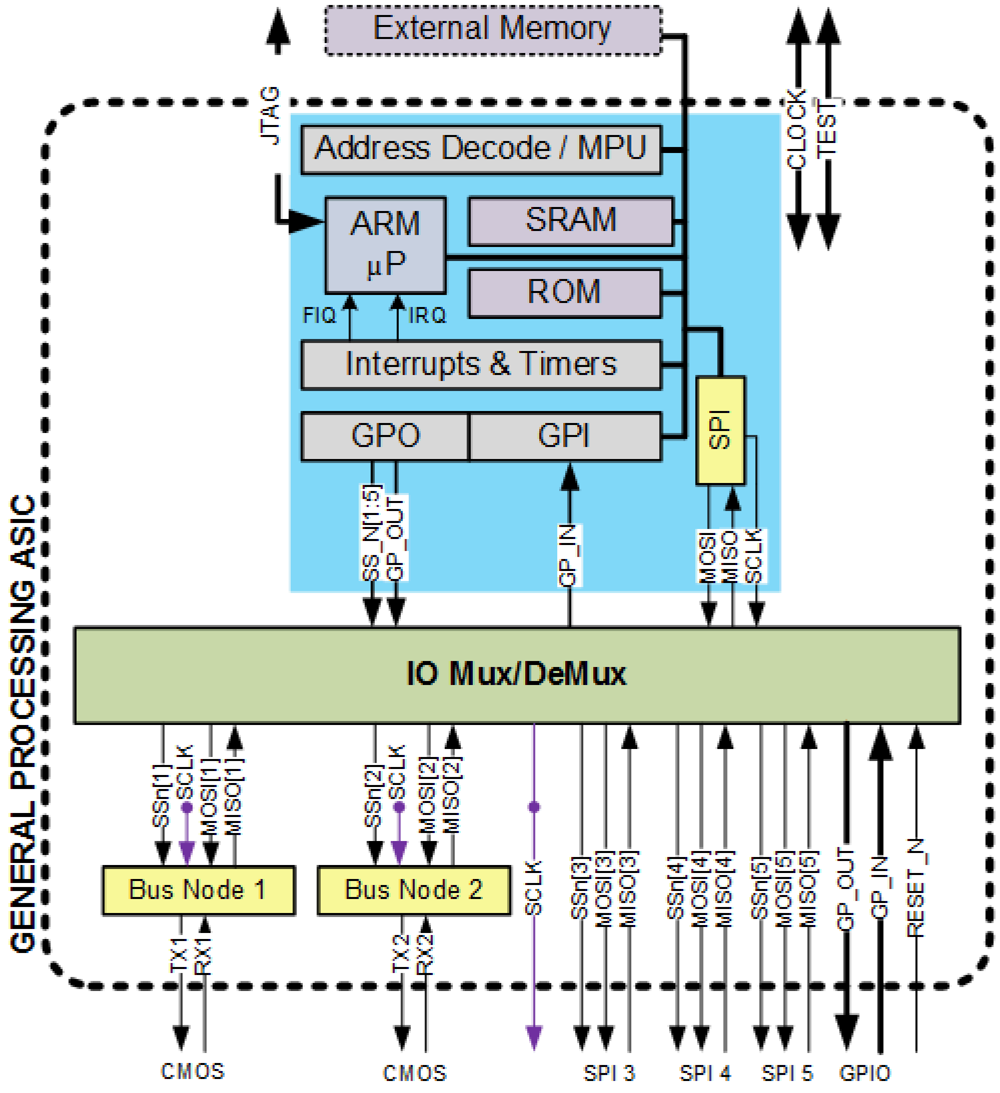
Albuquerque, New Mexico 87185 USA

Physical Device Verification is Challenging

- **In-system ASIC verification is complicated**
  - Board parasitics
  - Device delays
  - Clock skew
  - Signal impedance differences through connectors
- **Goal is to achieve the same level of test coverage during physical device verification as provided during constrained random simulations**
  - Testing a wide variety of combinations of interface parameters using directed tests can be time consuming, expensive, and ineffective
  - Extending the methodology of constrained random simulation into physical device verification reduces the number of test vectors required to achieve a high level of coverage

Sandia National Laboratories

General Processing ASIC (GPA) Block Diagram



Straightforward Test Process

1. Set up / execute constrained random simulations to generate the randomized value sets
2. Convert the simulation output to debugger commands for loading randomized value sets into memory
3. Develop test code to run on the GPA processor
4. Run the randomized tests in hardware

Sandia National Laboratories

Test Process Outputs

Step 1: Randomized Value Sets

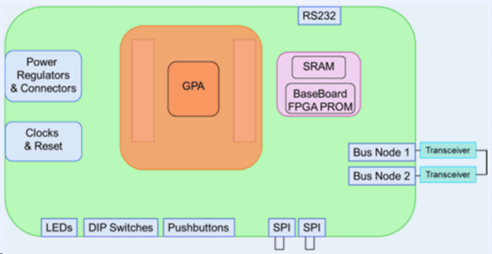
```
Clock Divide      CPOL      CPHA      Length      Data
---------------------------------------------------------
    8286            1         1        7993        241
    8286            0         0        7993        241
    8286            0         1        7993        241
    8286            1         0        7993        241
    6320            1         0        5791        107
    6320            0         1        5791        107
    6320            0         0        5791        107
```

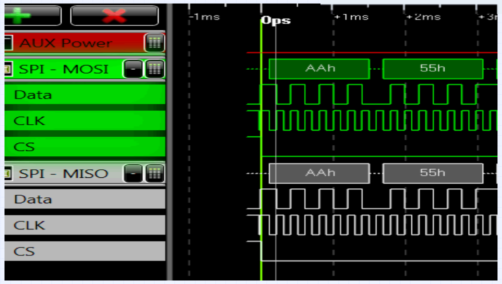
Step 2: Debugger Commands

```
memwrite 4 0x10011000 520
memwrite 4 0x10011004 8286
memwrite 4 0x10011008 1
memwrite 4 0x1001100c 1
memwrite 4 0x10011010 7993
memwrite 4 0x10011014 241
```

```
for (i=1; i<= numDataBlocks; ++i)
   {    dataBlock.clockDiv = *ptr; ptr++;
        dataBlock.cpol = *ptr; ptr++;
        dataBlock.cpha = *ptr; ptr++;
        dataBlock.length = *ptr; ptr++;
        dataBlock.data = *ptr; ptr++;
        loadBuffer(transmitBuff, dataBlock.data, numDataBytes);
        mode = dataBlock.clockDiv <<16 | dataBlock.cpol <<14 | dataBlock.cpha <<13;
        SPI_Open((unsigned int*)transmitBuff, dataBlock.length, mode, int_SPI);
        SPI_Transmit((unsigned int*)receiveBuff, dataBlock.length);
        if(compareBits(transmitBuff, receiveBuff, dataBlock.length) == FALSE) {
          SetOut(GPO_ERR);
        }
}
```

Step 3: Processor Test Code





Step 4: Hardware Test Results

Languages / Methodology Used

Process Step	Language / Methodology
Random Simulations	Open Source VHDL Verification Methodology (OS-VVM)*
Simulation Output Conversion	Python 3.0
Processor Test Code	C

* OS-VVM is an intelligent test bench methodology that allows mixing of coverage-driven randomization with common test approaches (directed, file based, and constrained random). For more information, visit **http://osvvm.org/**

Process is independent of the specific tools used

Sandia National Laboratories

Step 1: Execute Constrained Random Simulations

- **Identify the interfaces to be randomized**
- **Determine the parameters of the interfaces, the range of allowed parameter values, and the number of coverage bins**
  - Parameters based on the GPA interface software drivers
  - Range of parameter values based on GPA software drivers and practical constraints
  - Number of coverage bins set by number of desired samples in each range
- **Develop the RTL to be simulated**
  - Include the processes/procedures to perform the constrained randomization
  - Add process to output resulting randomized value sets to text file
  - Loop through randomization processes until all bins achieve 100% coverage

GPA Interfaces to be Randomized

- **External SPI Port:**

Parameter	Range	Coverage Bins
Clock Divide	8 - 8330	130
SPI Mode	0 - 3	4
Transaction Length (bits)	1 - 8192	29
Data	0 - 255	13

- **Internal SPI Port to Bus Nodes:**

Parameter	Range	Coverage Bins
Node TX	0 - 1	2
Transaction Length (bits)	1 - 2032	32
Node data value	0 - 255	13
Destination ports	16 - 65535	27

Sample Randomized Value Sets

- **SPI Random Values**
  - **520 Random Value Sets**

Clock Divide	SPI Mode	Length	Data
8286	11 00 01 10	7993	241 14
6320	10 01 00 11	5791	107 148
1284	01 10 11 00	794	69 186

- **Bus Nodes Random Values**
  - **64 Random Value Sets**

Nodes 1&2 TX	Length	Data	Dest. Port 1	Dest. Port 2
01 10	72	196 59	18155	17708
01 10	77	125 130	22166	22200
10 01	105	255 0	27910	28634

Sandia National Laboratories

Step 2: Convert Simulation Output File to Debugger Commands

- **Define location of randomized value sets in processor memory map**
- **Determine debugger commands/process for loading processor memory**
- **Develop script to convert randomized simulation output file to debugger commands for loading processor memory**
  - Decided to have the simulation output a general text file instead of direct debugger commands for ease of reviewing randomized simulation results
  - Use of a different debugger only requires modification of the conversion script
  - Modifying the VHDL and / or re-running the simulation is not required

Step 3: Develop Processor Test Code

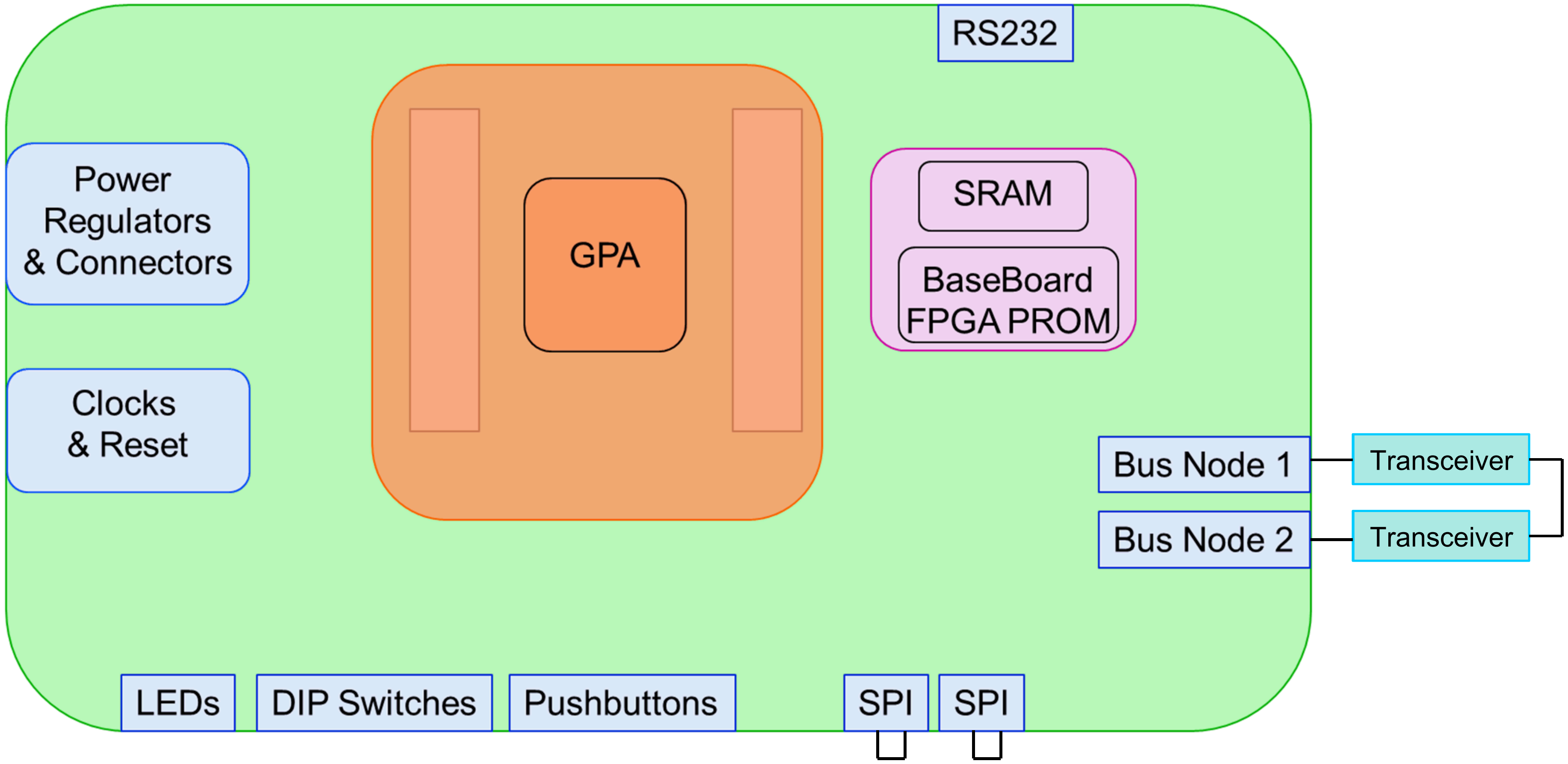
- **Number of randomized value sets is read from memory**
- **For each randomized value set**
  - GPA interface software driver is called with the value set
  - Results are compared to the expected results
    - » Output of each interface was looped back to verify that the data received was the same as the data sent

Step 4: Execute the Randomized Physical Test

- **Set up test platform**
  - Connect desired logic analyzers / scopes
- **Connect software debugger to processor**
- **Run debugger command file to initialize processor memory with randomized value sets**
- **Download the processor test code**
- **Execute test**
- **Monitor outputs and results**

Sandia National Laboratories

GPA Test Platform



Summary and Results

- Test process increases coverage during physical verification while decreasing required test time

Interface	Directed Value Sets	Randomized Value Sets	% Reduction
Internal SPI	22,464	64	~99%
External SPI	196,040	520	~99%

- Large investment in expensive tools is not required
- GPA hardware has successfully passed all system tests in three different applications that have varying interface requirements