FPGA Implementation of Advanced High Performance Bus Protocol for System on Chip Integration

Suseendran. K, Porutselvam. N
Department of Electronics and Communication Engineering, Gnanamani College of Technology, Namakkal, Tamil Nadu, India

Article Info
Article history:
Received 3 February 2015
Received in revised form
20 February 2015
Accepted 28 February 2015
Available online 6 March 2015

Keywords
Hardware Trojan Horses,
Reconfigurable Logic Device,
Dynamic Reconfiguration,
Dynamic Function Replacement,
System-on-Chip,
Advanced Microcontroller Bus Architecture

Abstract
A set of design methodologies and experiments related to enabling hardware systems to utilize on-the-fly configuration of reconfigurable logic to recover system operation from unexpected loss of system function. Methods included programming using locally stored configuration bit stream as well as using configuration bit stream transmitted from a remote site. A specific ways of utilizing reconfigurable logic to regenerate system function, as well as the effectiveness of this approach as a function of the type of attack, and various architectural attributes of the system. Based on this analysis, proposed architectural features of System-on-Chip (SOC) that can minimize performance degradation and maximize the likelihood of seamless system operation despite the function replacement. This approach is highly feasible in that it is not required to specially manage system software and other normal system hardware functions for the replacement.

1. Introduction

Design outsourcing has become increasingly common over the past 15 years for ICs generally and in particular for SOCs. The incorporation of third party IP designs represents an important potential point of vulnerability. Outsourced designs are typically provided using register transfer level (RTL) descriptions or hard macro cell designs with the result that there is no trusted golden model to use for comparison. In addition simulation models delivered along with third party IPs can themselves be untrustworthy and could be designed to block the modeling of the impact of an activation trigger. In such an environment, a trustworthy system-level model maybe difficult or impossible to obtain. Most fundamentally, a true Trojan would involve an attack designed to remain hidden and inactive until triggered either internally or externally and would be extremely difficult to detect during verification. In this regard, outsourced designs need protection against Trojan attacks.

2. Previous Work

The previous work relevant to the current paper lies in two broad areas: publications directly addressing hardware Trojans and those more generally addressing issues of reliability, self-repair and regeneration of circuits and embedded reconfigurable hardware. In the first category, there have been a number of publications that address methods to detect maliciously altered hardware *pre-deployment.* In [6], the authors have developed a scalable hardware Trojan detection and diagnosis method. The approach uses circuit segmentation and gate-level characterization to detect and diagnose hardware Trojan horses even in large circuits. Tehranipoor and Koushanfar presented a classification of hardware Trojans, and a survey and analysis of published techniques for Trojan detection in [7]. They also described recently proposed techniques used in designing for hardware trust. In [8] and [9], we described a set of anti-Trojan design methods and countermeasures that can increase IC security by making it possible to identify and quarantine a functional block found to contain a Trojan. Embedded reconfigurable logic has been used to implement function designs which may be updated in post deployment, unexpectedly added to systems after IC fabrication or temporarily used to perform IC testing before deployment.The Proposed architectural features of System-on-Chip that can minimize performance degradation and maximize seamless system operation despite the function replacement

Corresponding Author,
E-mail address: susee.ece71@gmail.com

3. DFR Sequence and Seamless System Operation

The on-the-fly replacement of a function creates some critical obstacles to seamless system operation. During the configuration, a bus master may attempt to access the function being configured because other system functions are not aware of the replacement activity. This obstacle is addressed by taking advantage of delayed response capabilities that are supported in most bus protocols (e.g., the HREADY signal in AHB bus specification). When a slave cannot serve a request immediately, the slave postpones the activation of the acknowledge signal and the accessing master needs to wait for the slave's response until the acknowledge signal is activated by the slave. Similarly, the DFR controller can delay any access of bus masters going to the replacing function during the replacement processes. This method allows other system functions to locally operate even during the configuration. The drawback of this method is that the delayed response locks bus operation and results in temporary halt of bus operation.

An architectural method called bus split is used to address the stopped bus transaction. This method allows the reconfigurable logic module to nullify the access on it so that other bus masters can utilize the system bus during the replacement. Fig.1 shows an AHB-based connection among a bus arbiter, master, and slave modules for the bus split operation.

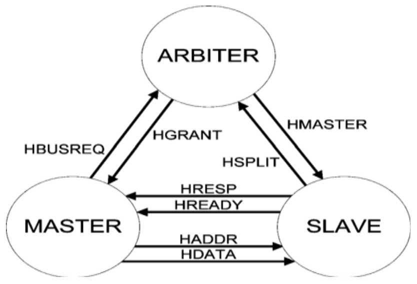

Fig: 1. AHB Based Connection among Bus Arbiter, Master and Slave for Bus Split Operation

4. Soc Bus Structure And Operation

System-on-a-chip (SOC) technology is the packaging of all the necessary electronic circuits and parts for a "system" (such as a cell phone or digital camera) on a single integrated circuit (IC), generally known as a microchip. A SOC consists of multiple heterogeneous functional blocks. To enable data flow among these blocks, a SOC bus is used to interconnect one or more processing cores to each other, and to the surrounding interface logic. This approach makes it possible to create functional blocks that are specialized for and therefore highly efficient at computation for specific tasks. Traditionally, SOCs are designed under the assumption that both the functional blocks themselves and the bus management logic are free from intentionally-inserted malicious circuitry. However, as SOC complexities continue to increase the number of vulnerabilities to corruption increases as well. In this environment, it is no longer possible to assume that a chip is always free of corrupted circuitry. Instead it is more prudent to design chips with the understanding that one or more blocks may prove to be corrupted and that on-the-fly identification and replacement of such blocks can be critical for the operation of the systems in which such SOCs reside. Thus in the present work, we uses SOC based AMBA bus to enable such replacement in the context of common bus architectures.

A. AMBA AHB BUS

The AHB takes on many characteristics of a standard plug-in bus. It's a multi-master with arbitration, putting the address on the bus, followed by the data. It also supports wait-state insertion and has a data-valid signal (HREADY). This bus differs in that it has separate read (HRDATA) and write (HWDATA) buses. These bus connections are multiplexed rather than making use of a tristate multiple connections.

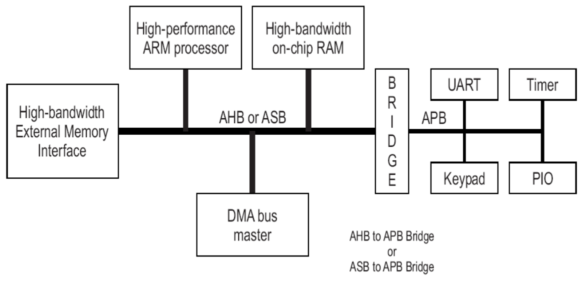
High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth External Memory Interface
AHB or ASB
DMA bus master
BRIDGE
UART
Timer
APB
Keypad
PIO
AHB to APB Bridge
or
ASB to APB Bridge

Fig: 2. AMBA Hierarchical Bus Architecture

All bus operations are initiated by bus masters, which also can serve as a slave. The master-generated address is decoded by a central address decoder that provides a select signal to the addressed bus slave unit. The bus master can "lock" the bus, reserving it with the central arbiter for a series of locked transfers. The slave unit has the option to terminate a transaction as an error, signals the master to retry, or split the transaction for later completion. Split transactions enable the slave to defer the operation until it's able to accomplish it thereby releasing the bus for other accesses. The slave signals a split transaction and saves the master number (HMASTER). When ready to complete the transaction, the slave signals the arbiter with the master number. When the arbiter grants bus access to the master, it restarts the transaction. No master can have more than 1 pending split transaction. AHB supports 32, 64 and 128-bit data-bus implementations with a fixed 32-bit address bus. It is a synchronous bus that supports bursts and pipelining of accesses to improve throughput. The AHB system bus and APB peripheral bus are linked through a 'bridge' that acts as the master to the peripheral bus slave devices.

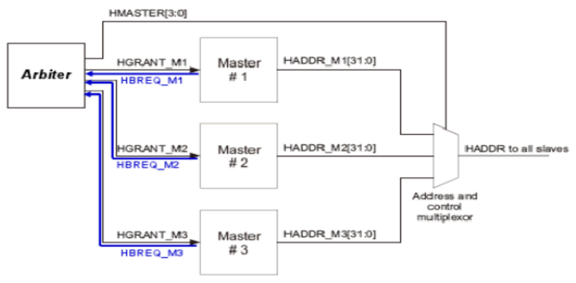

Fig: 3. AHB Arbitration

AHB supports multiple masters (either through a central arbiter, or through slave level arbiters in the case of a multi-layer AHB-Lite system). The arbiter has the task of determining which master gets to do an access. Every transfer has an address/control phase and a separate data phase. They're both pipelined (able to start the next transfer's arbitration and address phase while finishing the current transfer).The address transfer is always followed by the data phase. A slave (memory or peripheral device which accepts a read or write request from a master) can prolong the transfer (add wait states) using the HREADY signal. Separate unidirectional buses for read (HRDATA) and write (HWDATA) are used.AHB supports bursts which can either be of undefined-length or fixed length (4, 8 or 16 beats). Bursts may be performed to a fixed address (e.g. for wrap (where a critical word within a cache line is accessed first). The address from a master is decoded by a central address decoder that provides a select signal to one of the slaves. Slaves may respond to accesses by the master by signaling OK, or by reporting an error. In the full AHB system (but not AHB-Lite), slaves may also give a retry response, or the less commonly used split response. Split transactions let the slave to delay completion of the access until ready but to free the bus for other accesses by a different master. The slave records the number of the master and signals the arbiter when the split transfer can complete. When the arbiter re-grants the bus to that master, it restarts the transaction. A master can have only one pending split transaction.

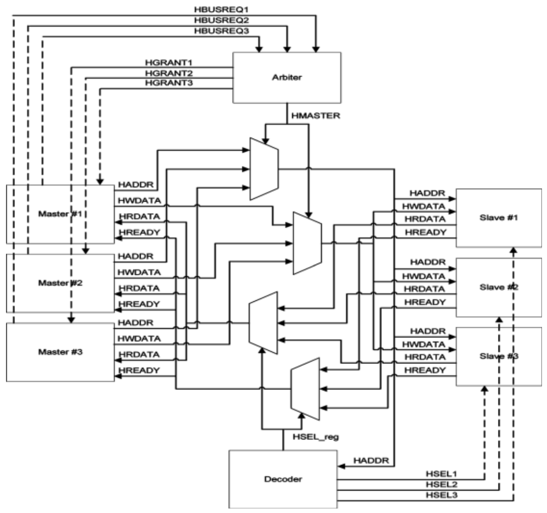

Fig: 4. A High-Level View of AMBA SOC Bus Interconnections, Showing Master and Slave Devices, an Arbiter, an Address Decoder and Various Multiplexers

B. AMBA BUS Arbitration

The AHB Arbiter is used in AMBA® 2.0 AHB multi-master systems to arbitrate the access to FIFO access), increment addresses (in steps of a single increment equal to the size of the access) or the AHB bus. The AHB Arbiter is basically a "traffic controller" which allows the AHB bus to be shared between multiple bus masters such as processors, DMA controllers, and peripheral core master interfaces. The AHB Arbiter uses a round robin priority scheme with Master0 having the default priority. This priority scheme assures that each master equally has its turn at acquiring and completing an AHB bus transaction. Each inactive master is locked out (HLOCK) while the active master has access to the bus to prevent contention. The AHB Arbiter steers all the AHB HWDATA, HADDR, HTRANS, HWRITE, HSIZE and HBURST signaling from each master to the AHB system bus. The AHB Arbiter is delivered as a three master arbiter but can easily be configured to allow up to sixteen AHB bus masters. IP Package the AHB Arbiter package includes fully tested and verified Verilog source. The AHB Arbiter can also be delivered as an FPGA Netlist for Xilinx and Altera FPGAs.

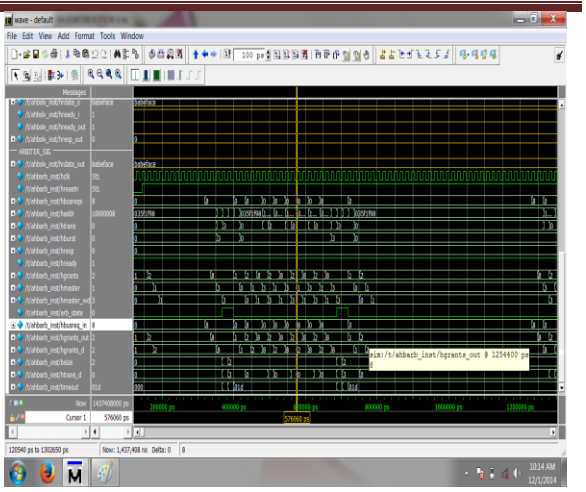
wave - default
File Edit View Add Format Tools Window
Messages
ARBITER_SIG
sim:/t/ahbarb_inst/hgrants_out @ 1254400 ps
Now
Cursor 1

Fig: 5. Simulation Result for Master

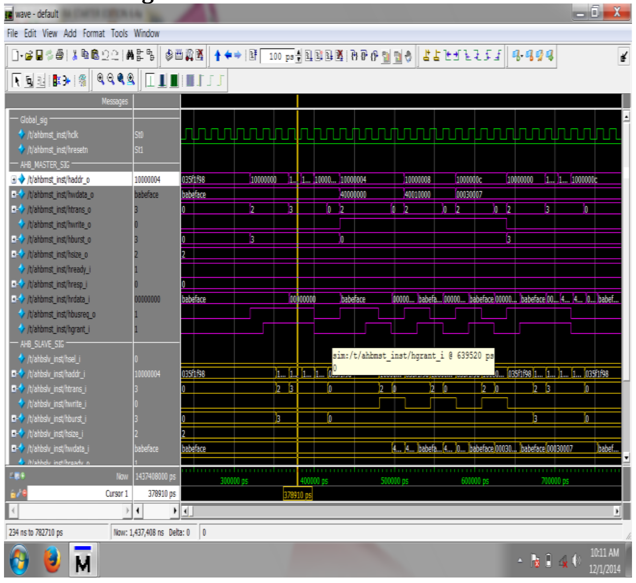
wave - default
File Edit View Add Format Tools Window
Messages
Global_sig
AHB_MASTER_SIG
AHB_SLAVE_SIG
sim:/t/ahbmst_inst/hgrant_i @ 639520 ps
Now
Cursor 1

Fig: 6. Simulation Result for Slave

5. Conclusions

We have described dynamic on chip replacement of functions disabled by Trojan IC attacks during run time, thereby enabling continued system operation despite the attack. This approach provides not only a way of regenerating compromised on chip function, but does so in a manner that enables seamless system operation during replacement and minimizes performance degradation.AHB protocol designed and measured in terms of reduced area, the cost to protect the Trojans in the system described here was found to be very modest.

References

[1] R. Saleh, S. Wilton, S. Mirabbasi, A. Hu, M. Greenstreet, G. Lemieux, P. Pande, C. Grecu, A. Ivanov, System-on-chip: Reuse and integration, in Proc. IEEE, 94(6), 2010, 1050–1069

[2] R. Rad, J. Plusquellic, M. Tehranipoor, Sensitivity analysis to hardware Trojans using power supply transient signals, in Proc. IEEE Int. Workshop

Hardware-Oriented Security Trust (HOST'08), 2008, 3–7

[3] Y. Jin, Y. Makris, Hardware Trojan detection using path delay fingerprint, in Proc. IEEE Int. Workshop Hardware-Oriented Security Trust (HOST'08), 2008, 51–57

[4] J. Aarestad, D. Acharyya, R. M. Rad, J. Plusquellic, Detecting Trojans though leakage current analysis using multiple supply padIDDQ, IEEE Trans. Inf. Forensics Security, 5(4), 2010, 893–904

[5] ARM, AMBA specification (rev 2.0), 1999