Design and Implementation of a System Bus – SPI Bridge for Wireless Radio Prototyping

Mohamad Yusri Mohamad Yusof, Devi Prasad, Smruti Santosh Palai
Microelectronics Department, MIMOS Berhad.
Technology Park Malaysia, Kuala Lumpur, Malaysia-57000.
myusri.myusof@mimos.my, devi.prasad@mimos.my, santosh.palai@mimos.my http://www.mimos.my

Abstract: - Prototyping of wireless radio is one of the major stages of the entire development process. It invariably has an interface to the Analog to digital converters (ADC) / Digital to Analog converters (DAC) and RF front end. This paper discusses in detail the design issues and solutions for baseband- RF front end interface, of a wireless radio. In these kinds of systems, the FPGA tends to be connected to the baseband DSP processor through the system bus. The FPGA is then used to implement additional baseband processing, hardware accelerators, ADC/DAC interfaces and RF control, to name a few. The DSP processor will need to communicate with these components through the system bus. Some of these components require high bandwidth and others requires low bandwidth from the system bus. There is a need for a design approach where that avoids slow interfaces, such as RF control interface, from hogging the system bus, which in turn will affect the overall performance of the entire system. This paper presents a System Bus-SPI bridge design approach to mitigate the interfacing issues in wireless system prototyping, especially when the supporting hardware, like RF module, is predefined. The proposed design enables the DSP Processor to access the System Bus concurrently while the SPI programming is in progress. Verilog hardware description language is used to design the System bus –SPI Bridge and Modelsim is used to verify the functionality of the design. The proposed design was implemented on an Altera STRATIX II FPGA.

Key-Words: - FPGA, SPI, Prototyping, System Bus, Wireless radio, Shared Bus.

1 Introduction

The rapid growth in demand for high data bandwidth stimulates the evolution of wireless technologies. This eventually pushes competing industries to rapidly develop high data rate and efficient wireless systems. As a result, rapid prototyping is becoming a vital part of the product development process [1] [2]. Prototyping is necessary not only to verify the developed algorithms but also to check the efficiency of the newly developed wireless radio [3].

Typical radio implementation consists of RF front end, baseband processing (or, PHY Layer), and MAC layer processing. In general, MAC layer processing is done by a general purpose processor (GPP). The PHY layer processing is performed by a digital signal processor (DSP), a Field Programmable Gate Array (FPGA), or a combination of both [4] [5] [6].

One common problem in prototyping the radio is in interfacing the baseband module with the RF frontend.

The final SoC design tends to have taken this interfacing problem into consideration. Sometimes it is difficult, however, to get development board that matches the architecture used by the SoC architecture exactly. As such, the interfacing problem still remains during the prototyping of the radio. It gets worse when the system bus is shared between a high data rate low latency access and slow transfers like SPI.

The RF frontend interface is based on the standard 4-wire SPI protocol [7][8][9] .In this paper we propose a bridge design to interface SPI ports to the RF frontend with the baseband processor through a shared system bus. This design has been verified on a prototyping platform (SPTWIMAXCC1E Multi-Standard Baseband AMC Channel Card) from Freescale Semiconductor [10].

This paper initially talks about the hardware prototyping platform. This is then followed by a brief description of the design to be implemented on the said hardware and the interfacing issues. To address these issues, the details of the design and simulation results of the System bus SPI Bridge and its implementation is explained towards the end.

2 Prototype Platform

The SPTWIMAXCC1E multi-standard baseband advanced mezzanine card (AMC) channel card is a system development platform for 4G wireless systems such as worldwide interoperability for microwave access (WiMAX) and wideband code division multiple access (WCDMA) markets. It is designed for use as a channel card module for a base station system solution or as a standalone platform for Pico-base station implementation. The platform is designed around the Freescale MPC8555E Power QUICC™ III processor, running at 833MHz, the StarCore MSC8126 multi-core

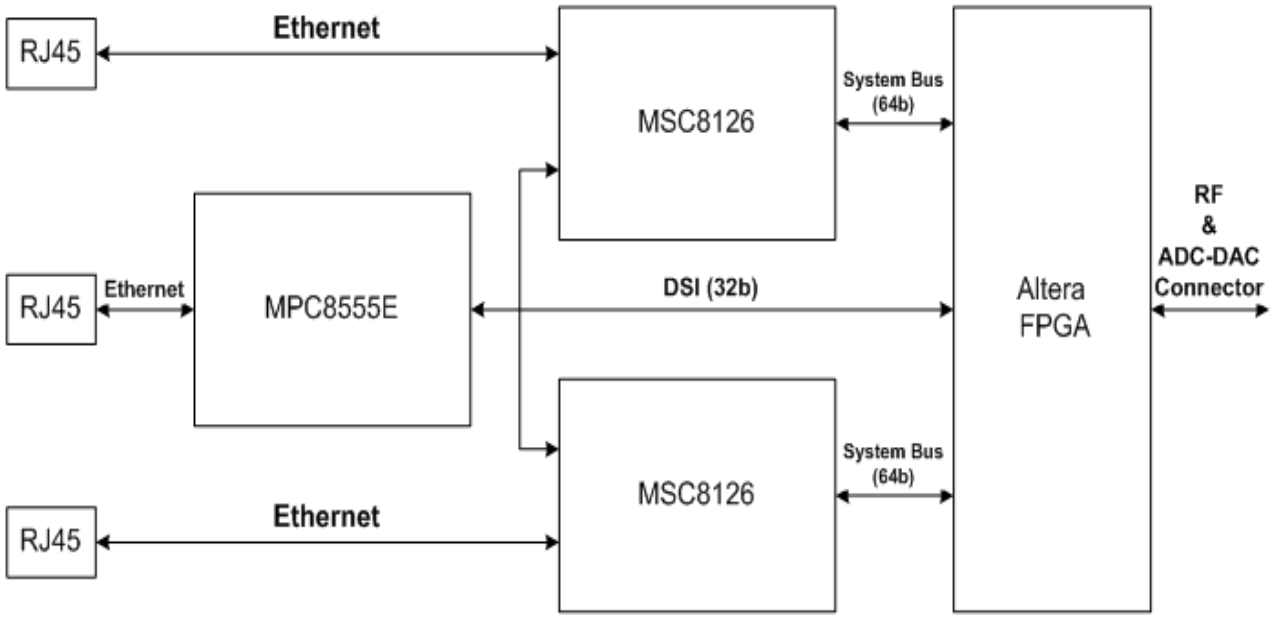

Fig.1 Freescale SPTWIMAXCC1E Platform

DSPs, running at 500MHz, and the Altera STARTIX II FPGA as shown in Fig.1. Generally, the MPC8555E is used for MAC processing while the MSC8126 DSP processors are used for PHY processing. The FPGA, which is used to interface with the RF frontend, ADC and DAC, is connected to both DSP processors on two separate system buses.

For a typical downlink processing, the received signal from the RF frontend sampled by ADC is passed to the DSP through the FPGA for PHY processing. This FPGA also serves as a hardware accelerator for the DSP to perform certain PHY functions as well. After the PHY processing is complete, the data is then passed to the MPC8555E for MAC layer processing and then is sent out through Ethernet port for user applications running at the host system.

2.1 MAC Processor (MPC8555E)

While the MPC8555E has many features, for example, features that interface with the AMC connectors, we will only describe features that are pertinent to the design described in this paper. An RJ45 port is available for 1 Gigabit Ethernet operation. It is used to pass packets from the MAC layer processing to a host computer for further treatment. The MPC8555E is connected to the two MSC8126 DSP processors and the FPGA through a 32-bit DSI (Direct Slave Interface) bus.

2.2 Baseband Processor (MSC8126)

The StarCore DSP subsystem consists of two MSC8126 DSPs running at 500 MHz to perform the symbol rate portion of the PHY layer processing.

The 64-bit system bus of each MSC8126 is connected to the FPGA which runs based on a 166MHz reference clock . Each DSP has 100 BaseT port connected to RJ45 or directly to the MPC8555E. DSI bus is accessible from MPC8555E local bus.

2.3 FPGA (Altera Startix II)

This FPGA is EP2S180F1508C3N (Stratix II Altera FPGA), which is used to time consuming processing at the PHY layer and other interfacing needs. It has 180K equivalent logic elements and 9Mbit on-chip memory and 450 MHz internal clock [11]. The FPGA is connected to each MSC8126 through a separate system bus and to the MPC8555E through the DSI bus. It also connects to RF module through an ADC interface. In addition, the FPGA also has a direct access to 512Mbytes of DDR2 memory for data storage.

2.4 RF frontend

A custom RF frontend board was designed for this prototyping. It has two 12-bit DACs to generate the baseband Analog signal from the digital samples received from DSP. The generated analog signal was up-converted using a zero IF RF transceiver chip for transmission. The received signal by the same transceiver chip was down-converted to baseband. This signal digitized by two 12 bit ADC in I/Q form. These I/Q samples are then forwarded to the DSP for baseband processing through the FPGA.

3 System Architecture

The targeted application was a prototype of a wireless broadband radio on the above platform. A portion of the

architecture of the system is shown in Fig.2 where the Receiver (RX) path of the radio has been depicted. The system consists of IQ sample collection from A/D convertor and the forwarding of the samples to the DSP without significant latency for further processing.

A portion of the PHY processing is done by FPGA. A RF transceiver module has been used for up/down conversion.

The RF module down converts the received signal in analog IQ form. Two A/D converters (one each for I and Q) digitize them and send them to the FPGA buffer for temporary storage. Subsequently, baseband processor (DSP) reads the stored IQ samples through System Bus Interface from FPGA.

3.1 Baseband Processing

The baseband processing algorithms are implemented on a MSC8126 DSP. The 64-bit System Bus, running at 166 MHz, is used to transfer I/Q samples to and from the FPGA. In addition, command and control signal for the RF frontend are also conveyed to the FPGA through the System Bus.

3.2 FPGA Interface

The Altera Stratix II FPGA is used for system interfacing and for some lower physical layer processing. The System Bus Interface, as shown in Fig.2, is a memory map decoder which enables the DSP baseband processor to read or write into the selected device or memory [12]. The FPGA also has a FIFO to store IQ samples temporarily before they get transferred to the DSP. This will allow for better system bus utilization.

3.3 System Bus Interface

The 166MHz system bus width can be up to 64-bit wide. However, depending on requirement on the bandwidth, the data bus width can be smaller. The system bus communication between MSC8126 and the FPGA is supported by either a User-Programmable Machine (UPM) or a General-Purpose Chip-Select Machine (GPCM) controller [13]. Although UPM allows for arbitrary waveform patterns to be defined, for example, to devices supporting burst access, GPCM was chosen in

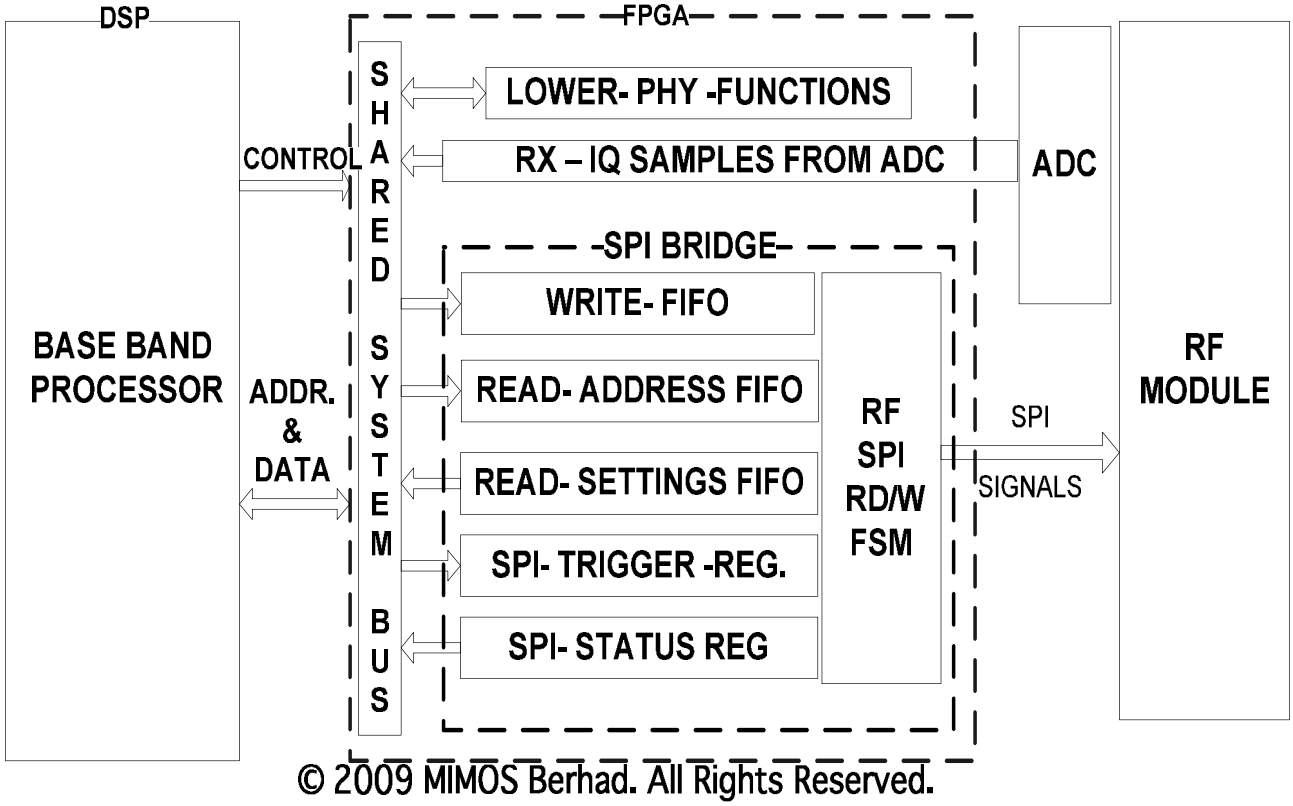

Fig.2 RX Path for the implemented radio (Partial.)

our implementation because it is simpler and it meets our bandwidth needs. In addition, a 32-bit data bus width was also selected.

GPCM interface of MSC8126 allows for a flexible and glueless interface between the MSC8126 and its peripherals. As shown in Fig.3, GPCM signals includes /CS (chip select), /PWE (write enables for write cycles), and /POE (read enables for read cycles). The GPCM contains two basic configuration register groups namely BRx, ORx. The entry in the BRx register selects the GPCM and the ORx defines the attributes for the memory cycles.

Fig.3 also shows the basic interface connection between MSC8126 and FPGA. Here /CS directly connects to /CE of the memory bank within FPGA. /PWE signals connect to the respective Write enable signals in the FPGA. In this design a two cycle read and write operation is performed by DSP to access the FPGA.

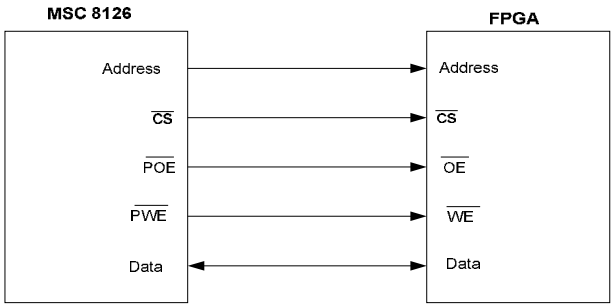

Fig.3 System bus interface between DSP and FPGA

3.4 RF Module

This RF module consists of a Zero-I/F transceiver chip. It requires 32 internal registers to be configured by a 4-wire SPI interface. These 4 wires are SCLK, CSB, DIN, and DOUT. SCLK is the SPI clock. CSB is active-low chip select signal, which needs to be low for SPI read or write operation. DIN is the serial input data and DOUT is the serial output data. For a typical write operation, SPI clock signal should be given to the SCLK input. Subsequently, the CSB signal needs to pull down to zero. The data to be written through SPI should be transferred serially on the DIN line with respect to the SCLK while SCB is asserted low.

The configuration registers are responsible for changing the operating frequency, gain and other RF parameters. One of the major parameters is the LNA gain register which needs to be updated through AGC loop to maintain the signal level within a valid range. These register requires frequent updating based on the values calculated by the DSP.

As shown in Fig.1. DSP is not directly connected to the RF module. The only option is for the DSP to access these RF registers is over the System Bus through the FPGA. Since the System Bus is also shared with high bandwidth I/Q sample transfer, as much as possible, the slow SPI transfer should not interfere with this crucial transfer. Hence, a System Bus-SPI bridge module has been developed to address this interface issue. This module enables the System bus to operate concurrently while the slow SPI programming under progress.

4 System Bus- SPI Bridge

A System bus-SPI bridge module has been designed using Verilog HDL, and the functionality verified using Modelsim-Altera. This is followed by the implementation using Quartus II. The design performance is then verified using on-chip debugger SIGNALTAP from Altera as well as real time downlink processing.

4.1 Design

The System Bus-SPI Bridge consists of three FIFOs and two registers and a FSM. This module was developed using Verilog HDL. It gives DSP access to 5 memory locations, namely; *rf_config_wr* (16-bit FIFO into which

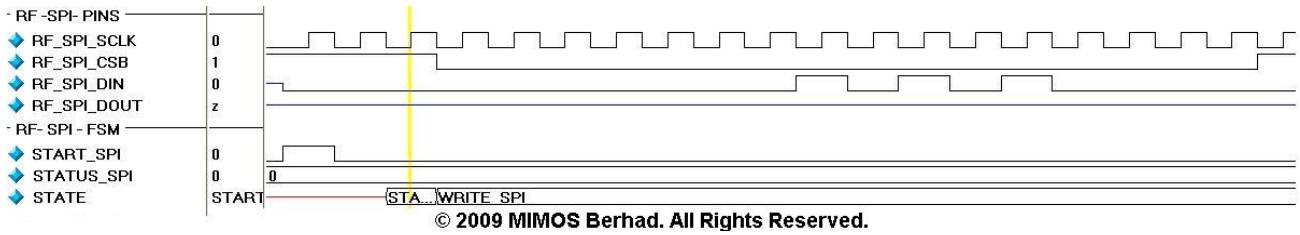

Fig.4 Write SPI.

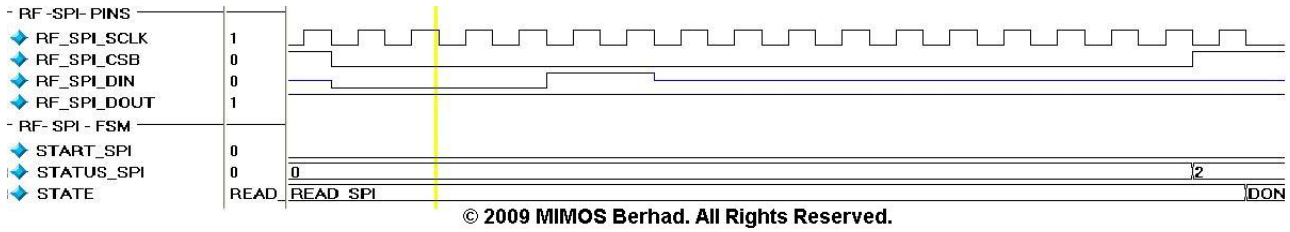

Fig.5 Read SPI.

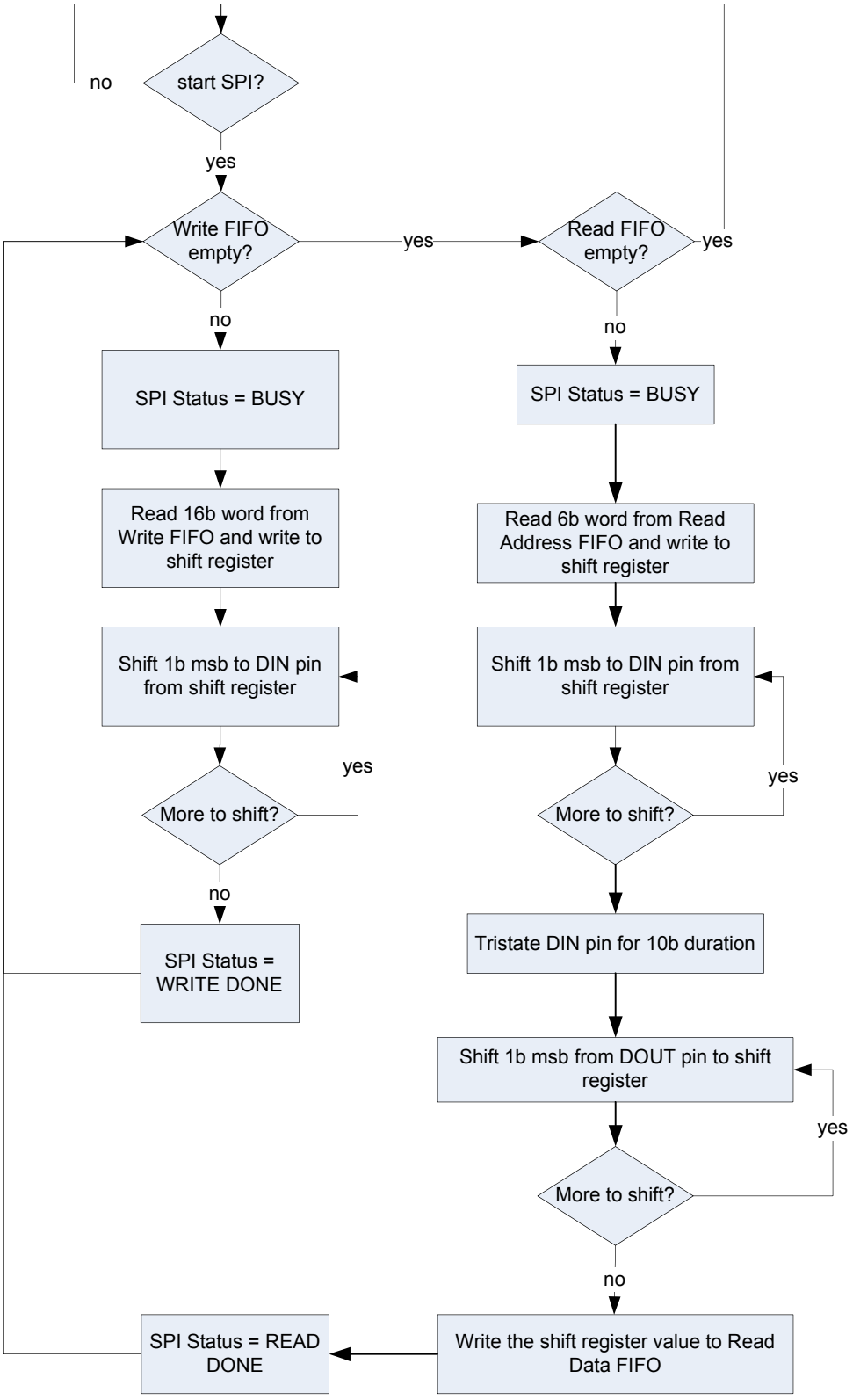

Fig.6. Flow chart for the Controller FSM.

address (6bits), parameter (10bits) to be written to the RF module through SPI),*rf_start_spi* (to start the SPI read/ write process), *rf_config_readaddr_wr* (6-bits address FIFO onto which DSP writes the desired address to read from RF module), *rf_config_read_addr_data* (16-bit FIFO from which DSP reads the values of the registers requested by earlier by *rf_config_readaddr_wr*), and finally *Rf_status_SPI (*2-bit register to show the status of the current SPI operation).

This bridge also connects to the RF module through the 4 SPI wires, namely, SCLK, CSB, DIN, DOUT. The waveforms of a typical SPI read and write operation shown in Fig 4 and Fig 5, respectively. Register data is shifted in MSB first and is framed by CSB. When CSB is low, the SCLK is active, and input data is shifted with the rising edge of the SCLK. Output data is used for read access and is shifted out to the registers in the falling edges of the SCLK.

The System Bus-SPI Bridge contains 32-word deep, 16-bit wide Write FIFO, to store the address (6 msbs) and configuration parameter (10 lsbs), which allow writing to up to 32 RF registers. In addition, the bridge has a 32-word deep, 6-bit wide FIFO to store the addresses of the registers for read back. The read-back values are stored in a Read FIFO, which is identical to Write-FIFO.

These read-back values are read by the DSP through system bus later.

A finite state machine (FSM) controller was designed to detect the DSP command and to control the SPI read and write operation. It also controls the data read and write to the corresponding FIFOs. Fig.6 depicts the flowchart of the FSM operations. The FSM detects the DSP command to initiate SPI operation, upon which it changes its state to “START”. At the next cycle, it checks for the empty status of the Read address FIFO and Write FIFO to distinguish between the read and write operation and changes the state accordingly.

If the Write FIFO is not empty it changes the state to “WRITE SPI”. Similarly, if the Read Address FIFO is not empty the state changes to “READ_SPI”. For write operation, first it sets the SPI-Status register value to busy. It then reads one 16-bit value from the write FIFO into a register. Next the register is shifted 1 bit at a time to the DIN pin for the RF module in MSB first order. After completion of all the shift operations it again checks for the FIFO empty status. The operations are repeated if it is not empty. Otherwise, it changes the state to “DONE” by changing SPI-Status register value to “Write done”.

For read operation, first it sets the Status register value to busy. It then reads 6-bit address from the FIFO into a register and shifts the register to the DIN pin of the RF module. It then presents the tri-stated value for next 10 clocks. The serial bits coming out of the DOUT pin of the RF module are stored in a 10 bit register. Finally this value is stored in the Read FIFO. This process continues repeatedly until the read address FIFO gets empty. Once the read address FIFO is empty, the state changes to DONE by changing SPI-Status register value to “Read done”

4.2 Simulation

The System bus operates at 166MHz and the SPI operates at 25MHz, which enables the DSP to use the system bus for other data transactions as well as the IQ sample transfers. To simulate the behavior of the developed bridge design a test model for the DSP had developed to excite the bridge design through system bus read and write. This model simulates the reading and writing of the actual DSP to the system bus and the bridge.

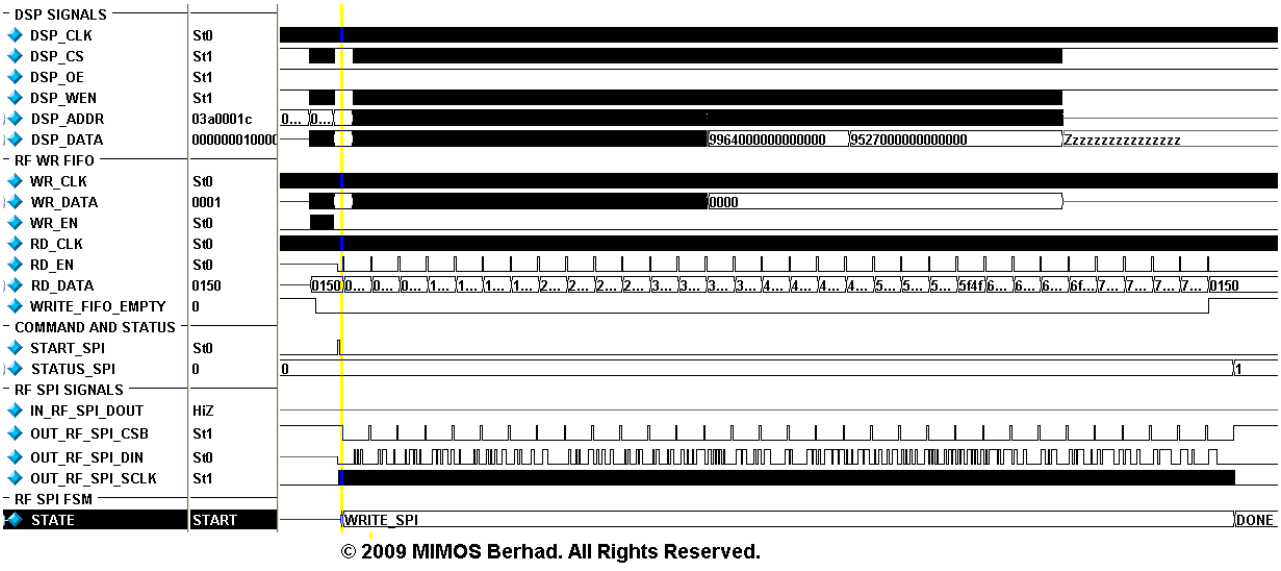

Fig.7 Top level simulation (SPI writes 32 registers & DSP using System bus to access hardware accelerator)

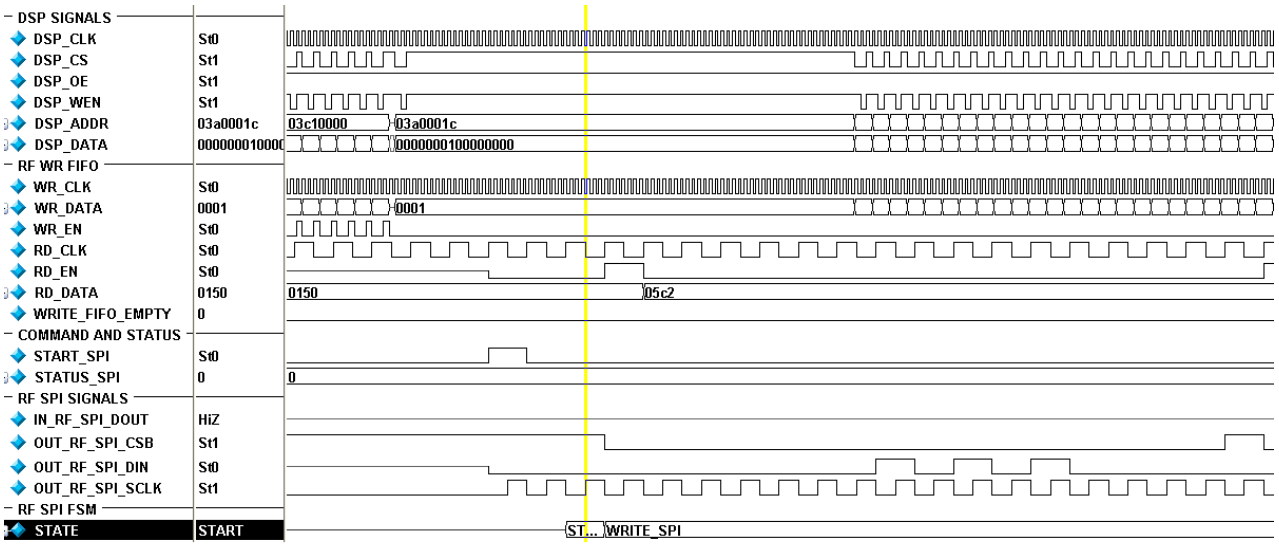

Fig.8 Top level simulation (SPI write & system bus Concurrent operation)

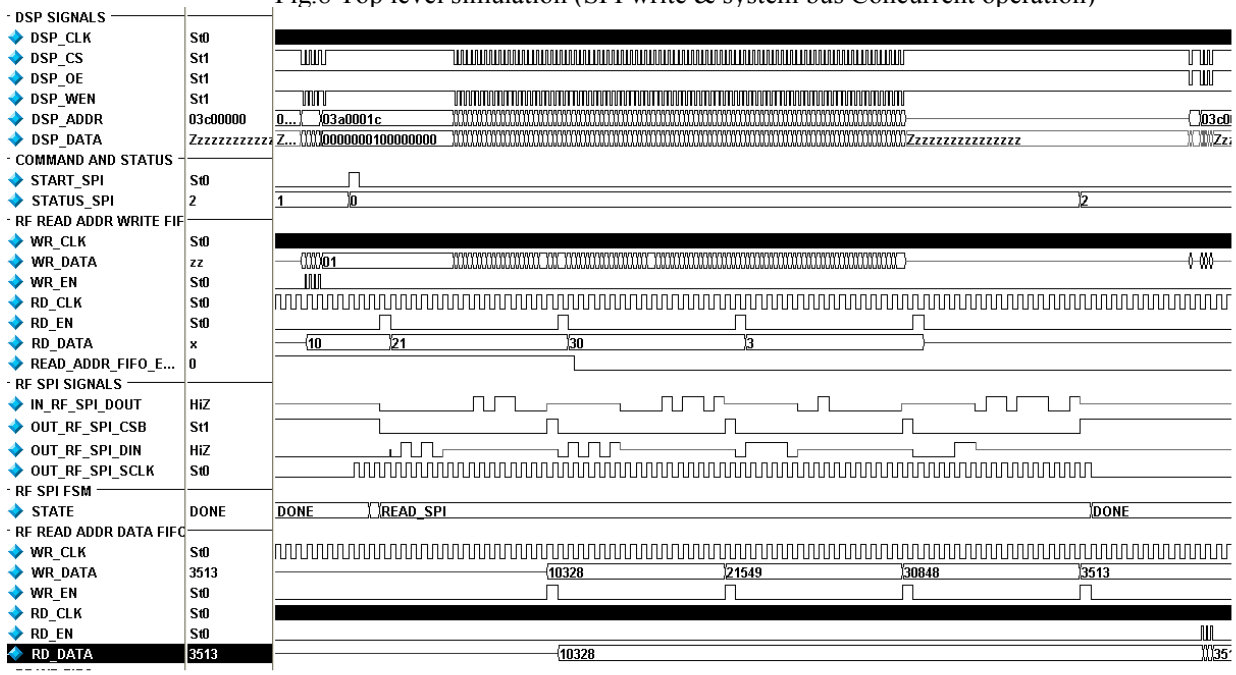

Fig.9 Top level simulation (SPI read & system bus Concurrent operation)

The functional simulation was performed in Modelsim-Altera. Fig.7 shows DSP model initially writes 32 register parameters to the write FIFO through the system bus at 166MHz. The DSP follows this with the *rf_start_spi* command write. The bridge receives the command "START SPI" and then checks the Read and Write FIFO status. Since the data is available in the Write FIFO, it starts the "SPI WRITE" operation. It continues the operation until the write FIFO is empty. Meanwhile, since the system bus is free during the SPI process, DSP schedules it to perform information transaction between the Lower PHY modules at the FPGA and the DSP at 166MHz.

It also shares it for the IQ transfer from the ADC. This improves the bus utilization significantly. A detailed view of the concurrent DSP write operation with the SPI write is shown in Fig.8. Fig.9 shows A Read SPI operation. Where DSP writes the address of the registers to be read to the *rf_config_readaddr_wr* FIFO and send the *rf_start_spi* command. The FSM controller detects this command and also detects the read address FIFO is not empty. Hence, it starts the READ SPI operation. Since the SPI operation is now independent of the System Bus, the bus is available for DSP to access the Hardware accelerator or to perform the IQ transfer.

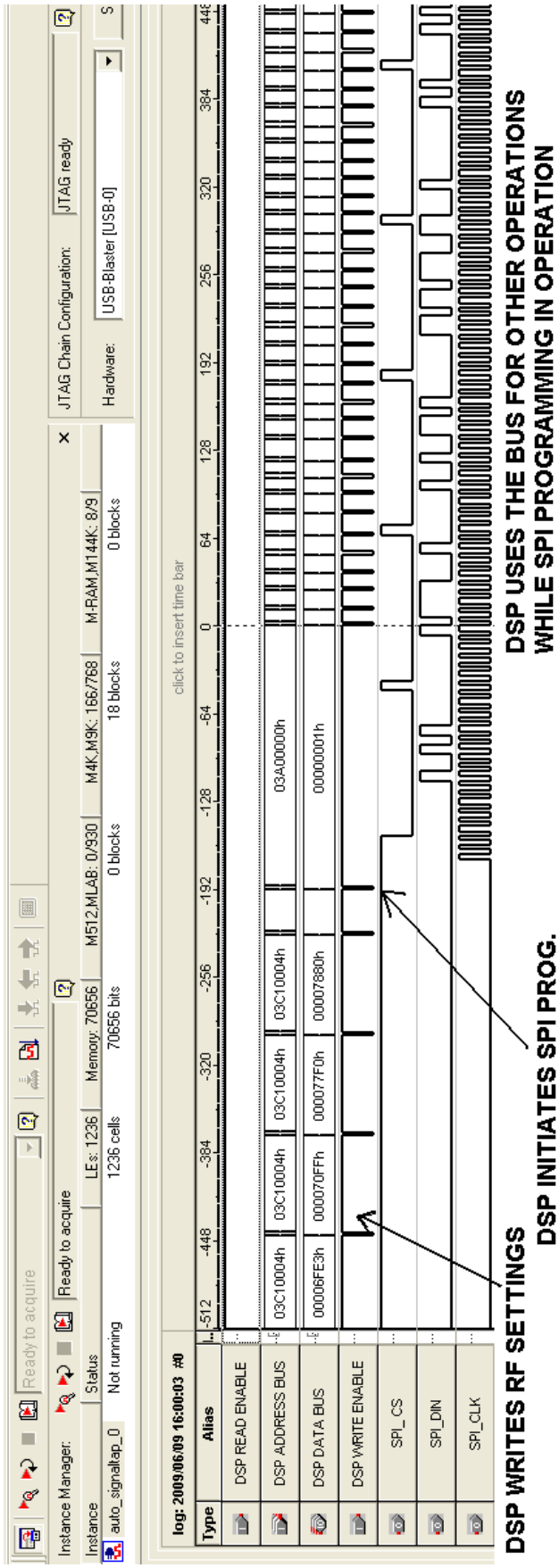

Fig.10 a SignalTapII waveform of a DSP write and SPI write concurrent operation

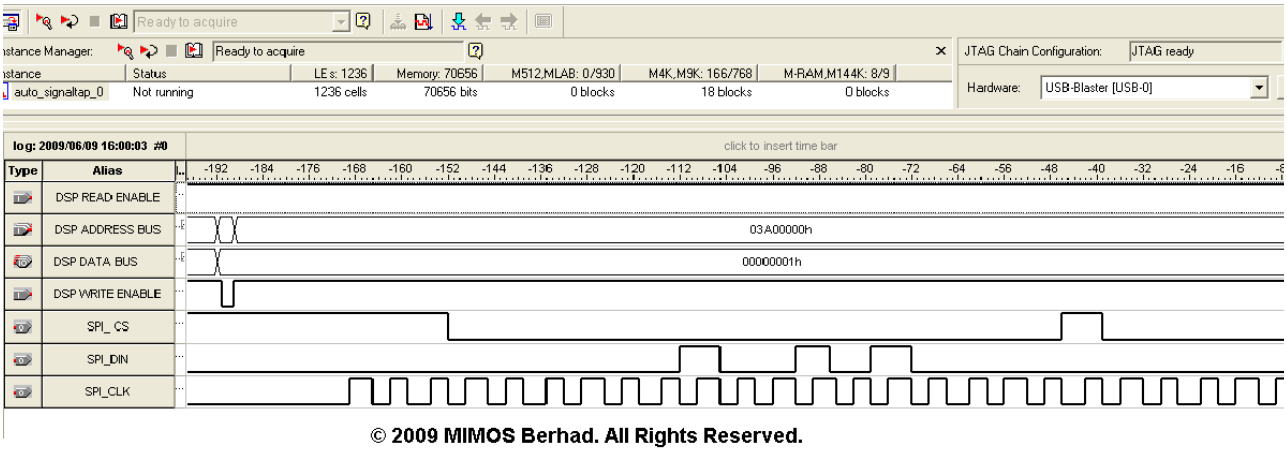

Fig.11 SPI Write

5 FPGA Implementation

The design was targeted to Stratix-II FPGA (EP2S180F1508C3) on the SPTWIMXCC1E platform. Quartus-II8.0 from ALTERA [14] was used for synthesis and implementation/fitting.

The real time performance of the design was verified by an on-chip debugging tool called "SignalTapII" [15], from ALTERA. With this tool the internal logic status was monitored real time.

The resource utilization of the system bus SPI Bridge was very minimal (283ALUTs, 351 Registers and 1,216 bits of Block Ram, as per Altera post fitting reports).

Fig.9 shows a SignalTapII captured waveform, in which DSP writes the RF settings to the FIFO through System bus. Then it initiates the SPI controller to start the SPI programming for the RF configuration. While the SPI programming is in operation, DSP uses the System bus for other data transactions between the PHY modules as the system bus is free. This is identical to the behavioral simulation results. Fig. 10 shows a SignalTapII captured SPI write waveform.

The design was successfully tested on the SPTWIMXCC1E platform and it was verified that the RF module could be configured at SCLK running at 25MHz without any timing violations.Fig.11 shows the post-fitt view of the implemented design.

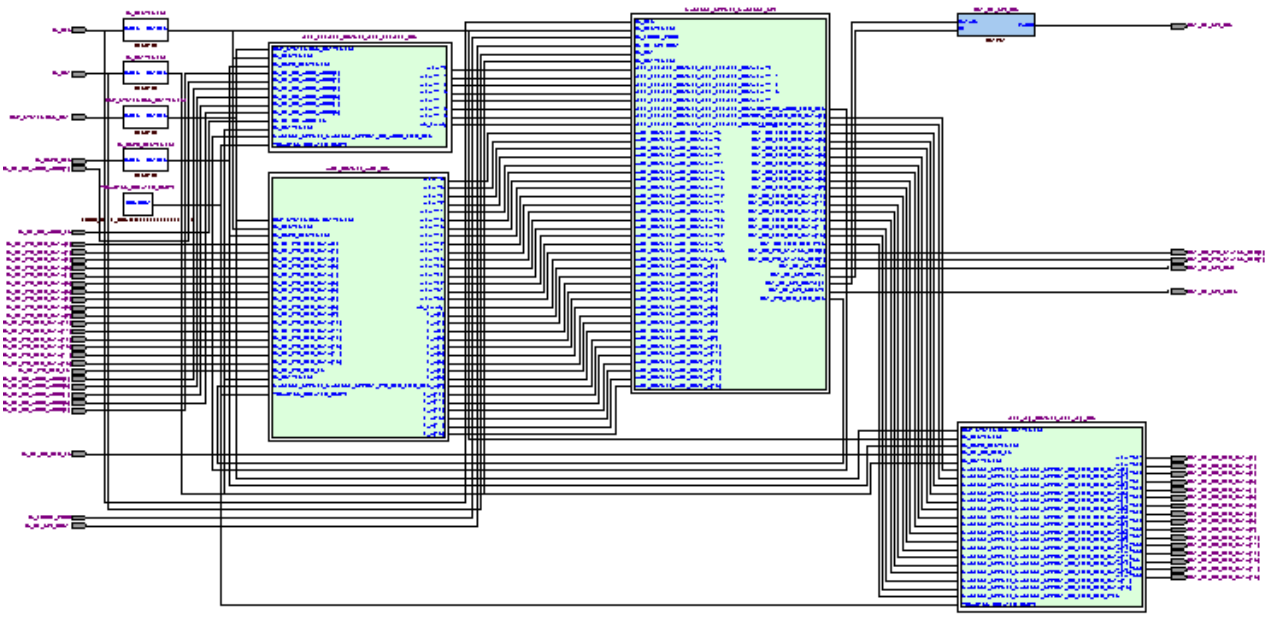

Fig.12 Post-Fitt view of the implemented bridge design

6 Conclusion

This paper explains the design of the System BUS-SPI Bridge to allow the DSP processor to configure RF module registers through the 4-wire SPI protocol. The design takes into consideration of the fact that the System bus is shared with other high-bandwidth transfers like I/Q data transfers. The bridge was designed is such a way that multiple slow SPI transfer may occur concurrently with other high bandwidth transfers. In short, the bridge design allows for efficient bus utilization. The bridge was developed and implemented in an ALTERA Stratix-II FPGA to overcome difficulties in interfacing the System Bus to a SPI complaint RF module.

References:

[1] J. Chen,W. Zhu, B. Daneshrad, J. Bhatia, H. S. Kim, K Mohammed, A Real Time 4x4 MIMO-OFDM SDR for Wireless Networking Research, *Proc. of European Signal Processing Conference*, *(EUSIPCO 2007)*, 2007, pp. 1131.

[2] S.M. Shajedul Hasan and S.W. Ellingson, Multiband Public Safety Radio using a Multiband RFIC with an RF Multiplexer-based Antenna Interface, *Software Defined Radio (SDR) '08, Washington DC,* 2008.

[3] Babak D. Beheshti, Performance Analysis of the WIMAX-D Physical Layer Blocks on a Next Generation Baseband Processor Platform, *12th WSEAS International Conference on COMMUNICATIONS, Heraklion, Greece*, July 2008.

[4] Aifeng Ren, Qinye Yin, FPGA Implementation of a W-CDMA System Based on IP Functions, *WSEAS Conf. on DYNAMICAL SYSTEMS and CONTROL*, Nov. 2005, pp. 320-324.

[5] Khaled Salah Mohammed, FPGA implementation of Bluetooth 2.0 Transceiver, *Proceedings of the 5th WSEAS Int. Conf. on System Science and Simulation in Engineering,* December 2006*,* pp.295-299.

[6] Ruei-Dar Fang,Hsi-Pin Ma, A DVB-T/H Baseband Receiver for Mobile Environments, *WSEAS Int. Conference on Circuits, Systems, Signal and Telecommunications,* January 2007, pp.114-117.

[7] Wen Xv, Hu Bing, Wang Wenbin, Design of Logic Control for Micro-power A/D with a Serial Interface Using FPGA, *The Eighth International Conference on Electronic Measurement and Instruments*, ICEMI'2007, pp. 4-870-4-873.

[8] Ieryung Park, Hosoon Shin, Jihan Park, Eungu Jung and Dongsoo Har, Improvement of TINYOS Implementation for Small Memory FPGA System, *XIII-IBERCHIP Workshop, IWS-2007*.

[9] Bacciarelli L., Lucia G., Saponara S., Fanucci L., Forliti M., Design, testing and prototyping of a software programmable 12C/SPI IP on AMBA bus, *Research in Microelectronics and Electronics2006*, RME2006, pp. 373 - 376.

[10] SPTWIMAXCC1E Multi-Standard Baseband AMC Channel Card, *Rev. 0,* April 2007, Freescale Semiconductor.

[11] Stratix II Device Family Data Sheet, *SII51001-4.2,* Altera Corporation.

[12] FPGA System Bus Interface for MSC81xx, *AN2823 Rev. 0*, August 2004, Freescale Semiconductor.

[13] MSC8126 Reference Manual, *MSC8126RM, Rev 2,* April 2005, Freescale Semiconductor.

[14] *www.altera.com*

[15] SignalTap II Embedded Logic Analyzer, *Quartus II Handbook.IV*, ALTERA.