The Design of MCU's Communication Interface

Borisav Jovanović, Dejan Mirković and Milunka Damnjanović
University of Niš, The Faculty of Electronic Engineering,
Niš, Serbia
{borisav.jovanovic, dejan.mirkovic, milunka.damnjanovic}@elfak.ni.ac.rs

Abstract— **In this paper, the communication between a microcontroller IP block and external Base band microprocessor is examined. The microcontroller is a part of a complex integrated System-on-chip and uses standard 8051 instruction set. The paper describes the operation of embedded circuits that allow programming, software debugging and communication with external microprocessor. The communication is based on SPI interface.**

Keywords - 8051 microcontroller, SPI interface, communication

I. INTRODUCTION

The 8051 microcontrollers are used in many electronic circuits and systems wherever some signal processing or a process control is required. Depending on the user requirements, many microcontrollers' characteristics have to be chosen, such as clock frequency and set of peripheral units.

There have been different implementations of the 8051 microcontroller [1], [2]. Also, there have been many examples of chips in which the 8051 microcontroller is implemented together with some other IP blocks [3]. In our case, the designed 8051 microcontroller IP block is part of an integrated power meter (IPM) System-on-chip [4]. The designed IPM incorporates all the required functional blocks for 3-phase metering, including a precision energy measurement front-end consisting of Sigma Delta AD converters, digital filters and digital signal processing block; 8051 microcontroller, real-time clock, LCD driver and programmable multi-purpose inputs/outputs. The IC requires a minimum of external components, inherently improving meter reliability, simplifying manufacturing process and providing a fast time-to-market metering solution. [4]

This paper describes one practically implemented system-on-chip, with communication between the integrated 8051 microcontroller block and the external Base band microprocessor. This paper thoroughly explains one efficient method for serial communication that allows microcontroller integrated block programming, software debugging and data transfer implementation.

In the Section 2 the global characteristics of the integrated microcontroller are given. Then, the microcontroller's interconnections to surrounding chips are explained. The serial communication interface which is used for basic control of the microcontroller is given. The operations of embedded circuits are explained that allow programming, software debugging and communication. After, in the Section 3, the microcontroller's implementation is described, which also explains the basic steps of verification phases within the integrated circuit design flow. Then, the suggested chip testing setup is described, which uses the serial communication routines for chip programming and testing.

II. THE MICROCONTROLLER

A. *The microcontroller's description*

The proposed microcontroller (MCU) IP block executes the industry standard 8051 instruction set. The instruction set is complex since it contains exactly 255 different assembler instructions. Moreover, the presence of six different addressing modes classifies this type of microcontroller as Complex instruction set computer (CISC). The microprocessors with complex instruction set are difficult for design, especially when tight design requirements have to be met, such as low power operation or high clock frequency.

Although the MCU has only 8 bits, because of the fact that the instruction set is supported by many software development tools, it is still very popular and widely used [5]. Also, the microcontroller fulfils the design requirements of System-on Chip (SoC) in which the MCU is incorporated, including the speed and benefits of rich instruction set.

The global architecture of proposed microcontroller can be divided into following blocks:

- the MCU core,
- the memory blocks,
- the block for programming and initialization
- the peripheral units.

The MCU core fetches, decodes and executes the instructions. It has the speed of one byte instruction executed in only two clock cycles. The maximum clock frequency is equal to 60MHz. The MCU offers the low power options, since it can operate at several clock frequencies. The MCU integrates the clock divider circuit which can reduce the clock frequency by a factor of 32.

The peripheral units include three 8-bit programmable digital parallel input/output ports, Inter-integrated circuit (I2C), Universal serial receiver/transmitter block (UART) and Serial protocol interface (SPI). The communication modules are used for the communication with external chips such as EEPROM memories and Base band processors. Furthermore, three counter/timer circuits are included.

All memory blocks, specific to 8051 microcontrollers, are physically implemented in the chip, including the 8kB SRAM used for program code storing. The 8kB memory block is volatile and program code is lost when chip is powered down. Instead of using on-chip non-volatile memory block, an external I2C serial memory is utilized. Namely, every time after the chip is powered up, the I2C memory content is read though I2C communication lines and loaded into 8kB SRAM memory block. Beside program code memory the chip incorporates two more SRAM memory blocks: Dual-port 256B internal RAM (IRAM) and 2kB external RAM (XRAM) blocks.

B. The MCU's interconnections to surrounding blocks

The microcontroller IP block is a part of complex SoC and it is used for autonomous control of other IP blocks which belong to the same chip - the digital filters, digital signal processing (DSP) block and real-time clock (RTC) circuit. Namely, the MCU is connected via local SPI lines (Fig. 1) to on-chip mixed signal blocks, which all have embedded SPI communication sub-modules with distinctive SPI addresses. In this communication the MCU is considered as a master unit, which initiates data transfer with other (controlled) IP blocks. The microcontroller's software simply read the status registers and writes the control registers of IP blocks via SPI.

The microcontroller itself is not fully autonomous. Instead, it is controlled by external Base band processor (Fig.1). Base band processor can reset the MCU or select one of the available options for MCU program code initialization. Namely, the MCU's program code can be loaded either from external I2C EEPROM memory or it can be loaded from Base band microcontroller via external SPI communication data lines (Fig.1). Beside the reset and programming options, the Base band can access the IRAM memory locations and SFR registers when MCU's software verification and debugging is performed. The communication with Base band controller will be further examined in detail.

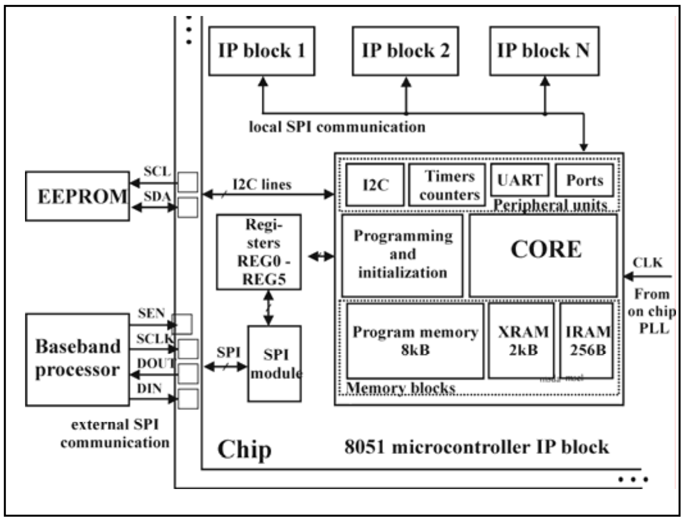

Figure 1. Global diagram of 8051 microcontroller, embedded into SoC

The MCU programming and control options are enabled by SPI module (Fig.1), which represents the main interface in the communication between Base Band processor and MCU. The SPI module utilizes the standard SPI communication protocol and following digital lines: SCLK – for the clock signal; DIN - input data line; DOUT – output data line and SEN – enable signal. The SCLK, DIN and SEN are module's input signals and DOUT is the output signal.

In the external SPI communication the role of MCU is changed compared to the internal chip SPI communication between MCU and other on-chip IP blocks. Now, the Base band processor is a master and the MCU can be considered as a slave. The main challenge during the chip design and verification is related to the Base band SPI clock (signal SCLK in Fig.1) which is not in synchronization with the main MCU clock signal. These two clock signals can have different frequencies and phases. Therefore, the synchronization between two clock domains had to be implemented too.

The SPI module reads and writes the 8-bit registers named with REG0 to REG5. These registers are used for MCU initialization, control and data transfer. For example, the registers REG0 and REG1 are connected to the parallel data input/output ports P0 and P1 which are used for the communication between the Base band and MCU. Namely, two MCU peripheral ports P0 and P1 are 8-bit wide and they are used to expand communication possibility between Base Band processor and MCU. The data at port P0 input is changed by writing data into the REG0 register; the Base band can read the port P1 output by reading the REG1 register.

Through REG2 the Base band controls the MCU's control input pins such as the main reset pin and programming mode selection input pins. Over REG3, the MCU provides the Base band its status signals. The REG4 and REG5 are used for data transfer between the MCU and Baseband which is needed during MCU programming, verification and testing procedures. The names, corresponding register addresses and short descriptions are given in the Table 1.

TABLE I. SHORT DESCRIPTION OF REGISTERS REG0-REG5

mSPI's register	register address	Description
REG0	0000000	Writes to port P0 inputs
REG1	0000001	Reads from port P1 outputs
REG2	0000010	Controls MCU input control pins
REG3	0000011	Reads MCU status signals
REG4	0000100	The data byte to be written into MCU
REG5	0000101	Reads data byte from MCU

C. The SPI registers used for data transfer between the MCU and Base band processor

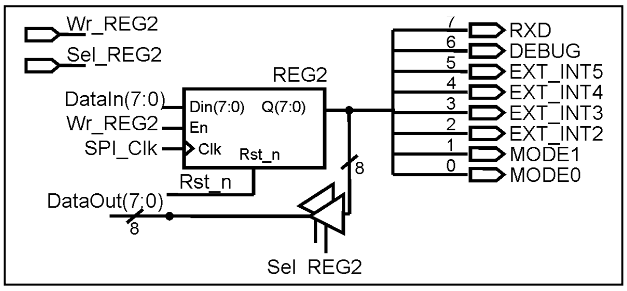

Figure 2. The register REG2 content controlling MCU input pins

The brief description of most important bits of registers REG2 and REG3 which are necessary for 8051 MCU communication with the Base band processor is given as follows.

The register REG2 (Fig. 2) is connected to MCU's input pins. The bits MODE1 and MODE0 of REG2 (bit positions 1 and 0) are used to control the chip boot-up. The detailed descriptions of chip programming modes will be given in the next Section.

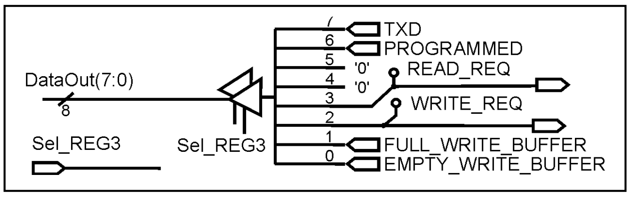

Figure 3. The register REG3 reading the status signals

The REG3 (Fig. 3) is read-only register. The bit 3 – the READ_REQ indicates that 8-bit data, available in the register REG5, is ready for transfer from MCU to the Base band. The bit 2 - WRITE_REQ is status signal holding the information that a new data byte in REG4 is waiting to be read by MCU. The FULL_WRITE_BUFF at bit position 1 indicates that 32-byte microcontroller's input FIFO buffer is full, so, Base band processor has to wait. The signal at bit position 0 - the EMPTY_WRITE_BUFF tells that 32-byte input FIFO is empty.

D. The read and write operations and the synchronization between the two clock domins

The REG4 write operations are performed during program code transfer from Base band to MCU and MCU's software debugging procedures.

During a write operation a data byte is written into the REG4. The following control signals are used: WRITE_REQ, EMPTY_WRITE_BUFF and FULL_WRITE_BUFF.

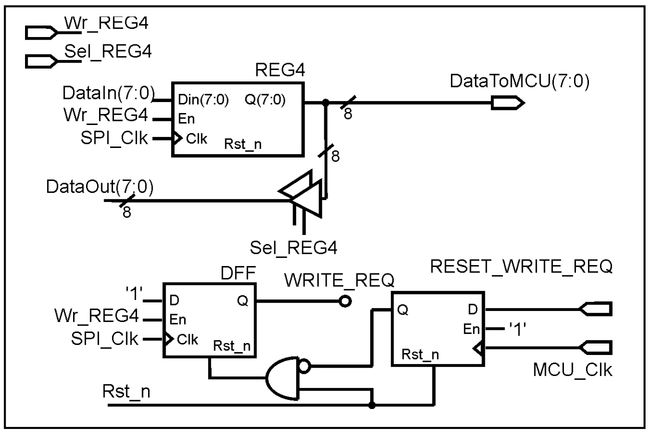

Figure 4. The data transfer to MCU through the register REG4

Since Base band's SPI clock and MCU's clock are not synchronized, the handshaking is implemented by signal WRITE_REQ (Fig. 4). The synchronization between two clock domains is achieved at the flip-flop producing the signal WRITE_REQ. The WRITE_REQ is set when new data byte is written into the register REG4. Namely, the WRITE_REQ is set to logic 1 synchronously with Base band's clock signal (the signal SPI_Clk given in Fig.4). After the MCU takes the data byte, it automatically resets the WRITE_REQ. The MCU's control logic which operates at clock frequency MCU_Clk, now asynchronously resets the flip-flop producing the signal WRITE_REQ.

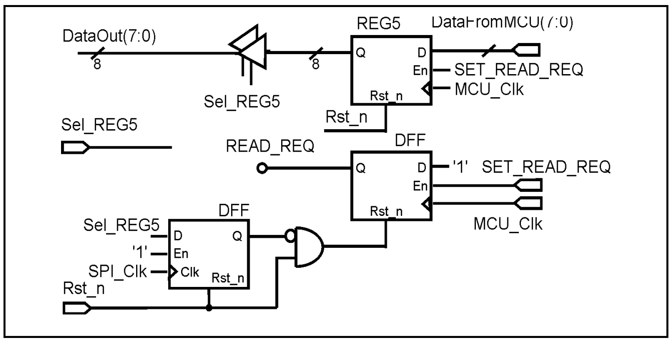

Figure 5. The data transfer to the Base band through the register REG5

The read operations happen during MCU's debugging. Then, the data is read by Base band through register REG5. Since Base band SPI clock and MCU's clock do not have to be synchronized the handshaking is implemented. The circuit implementing the read operation is given in Fig. 5. The synchronization between two clock domains is achieved at flip-flop producing the signal READ_REQ. When data byte is loaded into the REG5, the MCU sets the READ_REQ signal. Note that the signal READ_REQ is set synchronously with main clock signal of the microcontroller (the MCU_Clk given in Fig. 5). When REG5 is read through SPI module, the READ_REQ is automatically reset by SPI interface control logic. The control logic loading the register REG5 operates at Base band's clock frequency (signal SPI_Clk given in Fig. 5) and generates the short pulse which asynchronously resets the flip-flop producing the signal READ_REQ.

Synchronizing between clock domains is accomplished by registering the signals through a flip-flops that are clocked by the source clock domain, thus holding the signal long enough to be detected by the higher frequency clocked destination domain. To avoid issues with meta-stability in the destination domain, a minimum of 2 stages of re-synchronization flip-flops are included in the destination domain.

III. THE IMPLEMENTATION OF THE MICROCONTROLLER

A. Design and verification

The microcontroller was implemented in TSMC 65nm technology [6]. It operates at voltage supply of 1.2V. The following Cadence tools [7] have been used during the chip design:

- RTL Compiler for logical synthesis
- SoC Encounter for implementation.
- NCSim for logical verification

The MCU has the following advantages:

• new architecture provides speed of one 8-bit instruction executed in only two clock cycles

• operating frequency maximum is 60MHz

• low power operation; at frequency of 60MHz the power consumption is only 3mW.

The logical verification procedures focus on the MCU programming options which are achieved through SPI interface.

The MCU logical verification process begins by writing the C programs, which are compiled by Keil [8] or SDCC [9] into 8051 .hex file. After, the special C program converts the .hex file into VHDL code. This code provides the program memory content, which is instantiated into the main test bench.

The test bench program simulates the MCU's surroundings. The same test bench includes the instance of MCU which is being verified, the SPI communication module of Base band processor and external I2C EEPROM memory. The VHDL description of MCU also includes the VHDL code of internal SPI module. The control logic of test bench simulates the operation of Base band processor and describes the program code transfer to the MCU and EEPROM memory. The data bytes are sent first via SPI interface and after to the EEPROM.

During the logical verification process, the SPI module of Base band reads the program memory content and byte-after-byte transfers it to the SPI module of MCU. The logical verification results, obtained by simulation waveforms, present the SFR register's content. The MCU operation is simply verified by comparing the expected and obtained results.

The absence of synchronization which exists between the Base band SPI clock signal and main MCU's clock signal made the design of communication modules more difficult. To overcome this problem, additional circuits for synchronization are added to the communication blocks which synchronize two clock domains. The verification process considered different combinations of clock signal frequencies. The MCU's clock frequency maximum is 60MHz and Base band SPI communication is 100MHz. The MCU's clock frequency value is decreased by clock divider circuit. The SPI clock of Base band is changed from 100MHz down to 1MHz.

Two options are supported here, one using external (off chip) EEPROM and another without external EEPROM.

Option A: Using external EEPROM

- Base band sets the MCU programming mode MODE(1:0)="01"
- Base band uploads 8KB into SRAM Program memory.
- While receiving 8KB, MCU flushes Program memory into the EEPROM.
- When transfer is finished, the MCU starts executing the code.

With this option, also, the following is enabled:

• If Baseband sets MCU programming mode MODE(1:0)="11", the EEPROM content is read and uploaded into 8kB SRAM. The SPI is not used for program code transfer

• When transfer is finished, the MCU starts executing the code.

Option B: No external EEPROM

• Base band processor sets MODE(1:0)="10"

• Base band processor uploads 8KB into Program memory via SPI registers.

• After receiving 8KB, the MCU starts executing the code.

Option A loads the on-chip SRAM and external EEPROM. The Base band controller sets the bits Mode(1:0)="01" of REG2. Then, the bytes are sent in 32B packets over SPI. After the chip has been programmed, the MCU sets the status bit Programmed, which is located in REG3.

The on-chip program code memory is loaded from external EEPROM when Base band sets the REG2 bits Mode(1:0)="11". After chip is reset, the program code bytes are automatically read from I2C EEPROM and loaded into the SRAM. The control logic sent the control codes to the MCU which performs the program code loading procedure via I2C lines. The programming is finished when status bit Programmed is set.

B. The 8051 MCU testing setup

The general test setup is depicted in Fig. 6. It is comprised of control application, installed on computer and Printed Circuit Board (PCB), specially designed for testing purposes.

The computer is connected to PCB over USB port. PCB includes also the Baseband processor and I2C EEPROM, dedicated to MCU program code storage (Fig. 6).

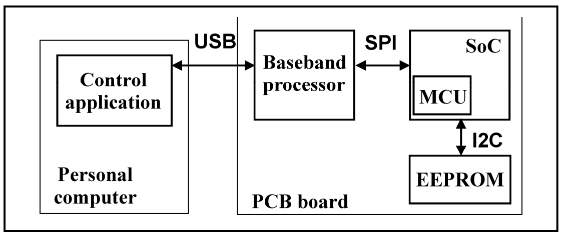

Figure 6. Global schematic of MCU testing setup

The control application implemented by [10], [11] sets the parameters of all subsystems within the chip. The part of control application is dedicated to 8051 MCU testing. It enables:

- The MCU reset
- HEX file loading, dedicated to MCU program code
- Selecting the MCU programming options A and B, as described below.

- Checking the MCU operation using Debug mode
- providing the IRAM memory and SFR registers content, which is valuable for MCU testing, and also, for MCU program code debugging.

The Base band firmware includes the software routines to read and write the MCU registers. These functions directly control the SPI communication lines between Base band processor and MCU. The algorithms which is used for MCU programming is described as follows.

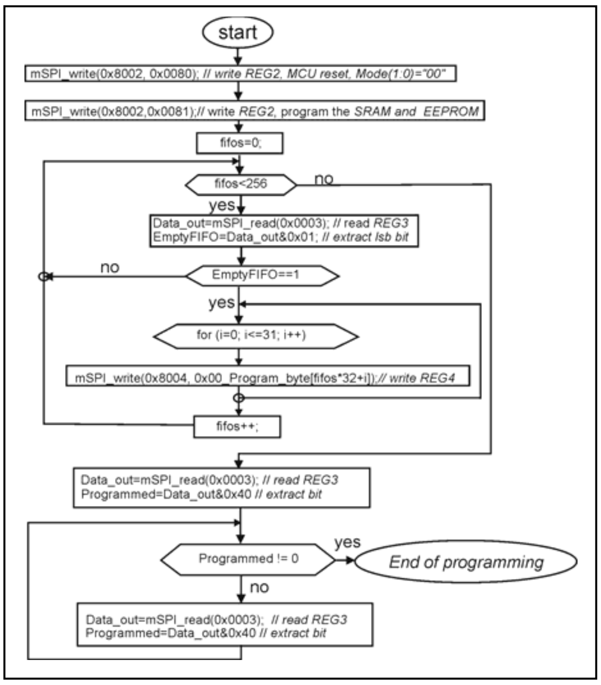
start
mSPI_write(0x8002, 0x0080); // write REG2, MCU reset, Mode(1:0)="00"
mSPI_write(0x8002,0x0081);// write REG2, program the SRAM and EEPROM
fifos=0;
fifos<256
no
yes
Data_out=mSPI_read(0x0003); // read REG3
EmptyFIFO=Data_out&0x01; // extract lsb bit
no
EmptyFIFO==1
yes
for (i=0; i<=31; i++)
mSPI_write(0x8004, 0x00_Program_byte[fifos*32+i]);// write REG4
fifos++;
Data_out=mSPI_read(0x0003); // read REG3
Programmed=Data_out&0x40 // extract bit
yes
Programmed != 0
End of programming
no
Data_out=mSPI_read(0x0003); // read REG3
Programmed=Data_out&0x40 // extract bit

Figure 7. The MCU programming operations

At the beginning, the MCU is reset by writing the command into register REG2 (bits MODE(1:0)=”00”). After, the bits are changed with MODE(1:0)=”01” or MODE(1:0)=”10”, dependent of the programming mode wanted. When programming mode 1 is chosen, the program code is written into both SRAM and EEPROM. When mode 2 is selected, the code enters only the SRAM. The program code is sent in data packets consisting of 32 bytes. The EmptyFIFO status signal gives the information when MCU’s FIFO is ready to take incoming bytes. The EmptyFIFO is checked by reading the REG3 register. The bytes are written into REG4. The operation is repeated 256 times to load 8kB MCU SRAM. At the end, the bit Programmed is checked.

IV. Conclusion

The 8051 MCU IP block is a part of a complex System-on chip and it is verified by systematic and thorough simulations. The results are confirmed after the layout was designed.

The microcontroller is connected to external Base band controller via SPI interface. In the serial communication, the MCU is a slave and the Baseband is a master. The MCU offers several programming options. In one of them, the program code memory is loaded from Baseband via SPI into both on-chip SRAM memory and external I2C EEPROM memory. Moreover, the Base band has control over MCU. The serial communication is particularly verified by simulations, the methods are described in paper.

The MCU’s main clock signal does not have the same frequency as SPI clock signal, which is produced by Base band controller. Therefore, the synchronization between different clock domains is done for reliable data transfer between the MCU and Base band.

Acknowledgment

This paper was funded by the Ministry of Education, Science and Technological Development, the project number TR32004, entitled with "Advanced technologies for measurement, control, and communications in the electricity grid."

References

[1] A. J. Martin, M. Nystrom, K. Papadantonakis, P. I. Penzes, P. Prakash, C. G. Wong, J. Chang, K. S. Ko, B. Lee, E. Ou, J. Pugh, E. V. Talvala, J. T. Tong and A. Tura, “The Lutonium: A Sub-Nanojoule Asynchronous 8051 Microcontroller”, in Proc. of ASYNC, 2003, pp. 14–23.

[2] J. H. Lee, Y. H. Kimand K. R. Cho, “A Low-Power Implementation of Asynchronous 8051 Employing Adaptive Pipeline Structure”, IEEE Transactions on Circuits and Systems II: Express Briefs, Vol. 55 , Issue 7, 2008, pp. 673 – 677

[3] K. L. Chang and B. H.Gwee, “A low-energy low-voltage asynchronous 8051 microcontroller cores,” in Proc. ISCAS, 2006, pp. 3181–3184

[4] B. Jovanović, M. Zwolinski, M. Damnjanović, “Low power digital design in Integrated Power Meter IC,” in Proc. of the Small Systems Simulation Symposium 2010, Niš, Serbia

[5] K. Arnold, Embedded controller hardware design, LLH Technology Publishing, Eagle Rock, USA, 2000.

[6] TSMC 65nm standard cell library, http://www.europractice-ic.com/technologies_TSMC.php?tech_id=

[7] Cadence, EDA software and verification tools http://www.cadence.com/us/pages/default.aspx

[8] KEIL C compiler and development tool for 8051 microcontrollers, http://www.keil.com/c51/ devproc.asp

[9] Open source SDCC C compiler for 8051 microcontrollers http://sdcc.sourceforge.net/.

[10] Code::Blocks IDE, http://www.codeblocks.org/

[11] GCC, the GNU Compiler Collection http://gcc.gnu.org