Online-Programming IP Design and Implementation Based on SoC

Wenrong Yang[1,2,*], Chao Jiang[1,2] and Yueli Hu[1,2]

[1]*Key Laboratory of Advanced Display and System Application, Shanghai University, Shanghai 200072, China*

[2]*School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, 200072, China*

Abstract: It is mainly proposed that the design and implementation for an online-programming IP core based on MV10, an SoC, which is an 8-bit MCU having passed the silicon verification with Global Foundry 0.35um mix-signal library in this thesis. The online-programming IP core can write BIN file into the on-chip SRAM, which is mainly controlled by erasing state machine and programming state machine, for program debugging with high accuracy-communication and error self-detection. The online-programming IP core proposed in this thesis can also be used in other SoCs.

Keywords: Online-programming, SoC, State Machine, High Accuracy-communication, Error Self-detection.

1. INTRODUCTION.

Generally, ISP (In System Programming) is mainly dependent on some external tools (except some parallel or serial programming devices) directly to download the program into a processor's internal program memory [1, 2]. Inserting the online-programming IP into the SoC can greatly improve the program debugging and execution. In this thesis MV10, a mixed-signal SoC, is used as the development platform for the realization of the online-programming IP core.

MV10 SoC is developed independently by the authors, which has full intellectual property rights of the ASIC [3, 4]. It is used in automotive electronics mainly for vehicle body control. The chip has the INTEL8051 kernel command structure, and embedded with PWM, ADC, CAN and other IP cores. MV10 is a silicon proven IP by chartered (now GLOBAL FOURDRY) 0.35um, 2 layer poly 4 layer metal mixed-signal process with 3.3v power supply. The operating frequency is up to 24MHz with 64K bytes of read-only memory and 256 bytes of internal data memory. MV10 has 4 input channels for 10 bits A/D converter, one UART and 5 timers/counters with 16-bit. It supports sleep/idle mode, and has 11 interrupt sources. It uses DIP52 package.

2. ONLINE-PROGRAMMING IP DESIGN

With having inserted the online-programming IP core into the MV10 SoC, PC can download any program into the program SRAM embedded in MV10 through the serial port. It makes MV10 have the ability of online debugging [5, 6]. Fig. (**1**) shows the total architecture of the online-programming system. It mainly includes SHU-MV Series MCU Assemble Program Development Software, Serial Port, Online-programming IP Core, Program SRAM and MV10 Main Core.

*Address correspondence to this author at the Key Laboratory of Advanced Display and System Application, Shanghai University, Shanghai 200072, China; Tel: 13918205699; 13501873485; E-mail: huyueli@shu.edu.cn

2.1. Top Level Design

The online-programming IP core mainly contains four sub-modules [7, 8]: sequence control module, data receiving module, data submitting module and SRAM interface management module. As shown in Fig. (**2**), it is the top structure of online-programming IP Core [9]. The pc_data_in is a signal for receiving the data from PC, and the pc_data_out is a signal for submitting the verification message to PC. The "Start" signal can switch the operation mode of MV10, that is, "0" for the download operation mode and "1" for the normal operation mode.

2.2. Sequence Control Module

The high accuracy of data communication is guaranteed by following three methods [10]: Two-level Synchronization, Frame-by-frame Calibration and Middle-Sampling. As shown in Fig. (**3**), after the Two-level Synchronization, the data flow "pc_data_in" are transformed into the stable data flow "pc_data_in_reg", thus it can effectively avoid the metastable. The third level synchronization of signal "pc_data_in_reg_reg" is used for the following "Start-bit" detection.

As shown in Fig. (**4**), in every rising pulse of the system clock, sequence control module will judge whether to begin the data sampling(bit_cnt[3:0] is a bit-counter for data-frame), according to the value of (!pc_data_in_reg & pc_data_in_reg_reg & (bit_cnt[3:0]==4 'b0)). After the detection of "Start-Bit", the system clock counter "clk_cnt" begins to work. So, the sample clock "clk_9600" will jump after "clk_cnt" count from "0" to "624" (The input data flow's baud rate set at 9600 Bit/s, the system clock is 12 MHZ, each bit of the data needs 1250 system clock cycle).

2.3. Data Frame Format

Communication data frame format between the online-programming IP and PC through serial port as shown in Fig. **5**. It includes one Start bit, 8-bit data, one Odd Check bit and one Stop bit. The data receiving and submitting module of

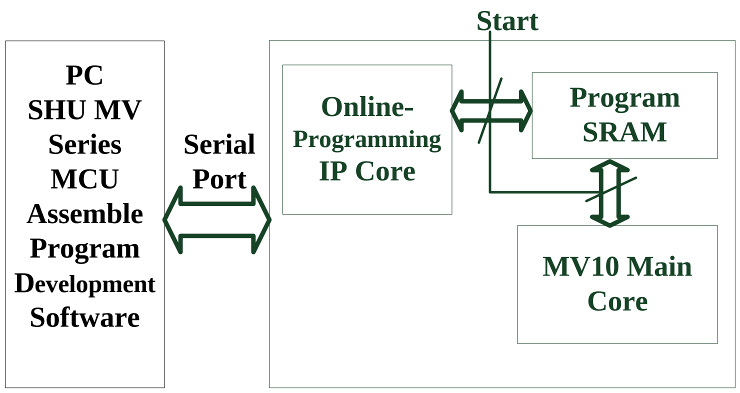

Start
PC SHU MV Series MCU Assemble Program Development Software
Serial Port
Online-Programming IP Core
Program SRAM
MV10 Main Core

Fig. (1). Online-programming System.

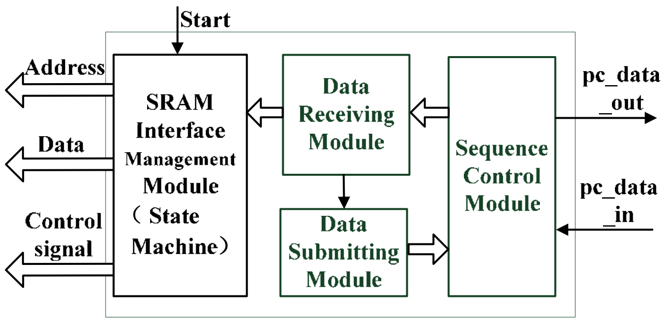


Fig. (2). Top Structure of the Online-programming IP.

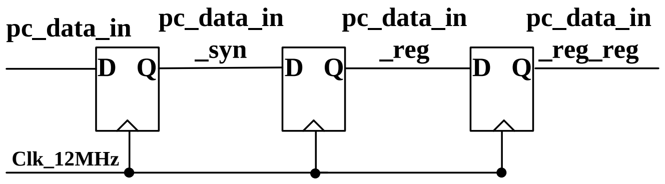


Fig. (3). Data Synchronization.

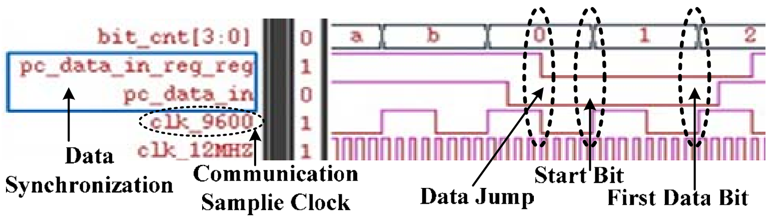


Fig. (4). Data Communication Sampling.

online-programming IP communicates with PC in accordance with this frame format.

2.4. Data Receiving and Submitting Module

When MV10 embedded with the online-programming IP is powered-on or reset, the data submitting module will automatically send “55H” to PC as a handshaking signal that it is ready to receive data. If the “55H” cannot be received by PC, it should be checked that whether PC’s UART is correctly configured, whether the MV10 is in the download operation mode, or whether the connection between them is correct. While receiving a data frame, the online-programming IP will check the data format, as shown in Fig (**5**), and then produce a flag according to the result checked, as shown in Table **1**. And, then it will send FFH or 00H to PC as shown in Fig. (**6**).

(1) Data Receiving

As shown in Fig. (**5**), the data frame receiving format includes a Start bit, 8-bit data, an Odd Check bit and a Stop bit.

Table 1. Feedback Data

Situation	Start Bit	Data Bit	Odd Check Bit	Stop Bit
Configuration Correct	0	55H	1	1
Data Correct	0	FFH	1	1
Data Error	0	00H	1	1

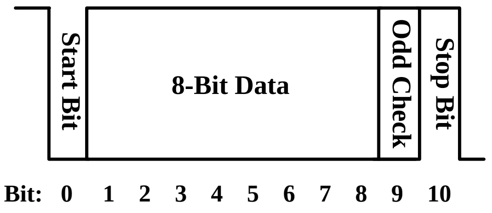


Fig. (5). Data Frame Format.

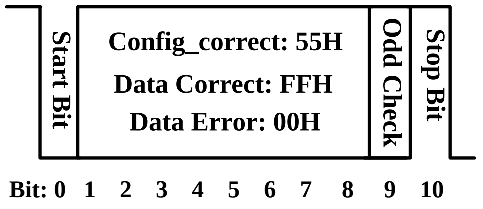


Fig. (6). Data Frame Submitting Format.

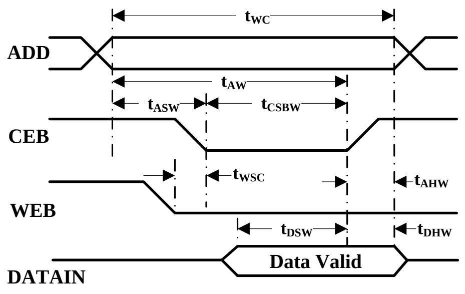


Fig. (7). SRAM Model Writing Time Sequence.

It extracts the data frame from the register according to the data packing format and writes the 8-bit data into the following program SRAM. And, Odd Check and frame verification for each data will be done to produce the error or right flag.

Error self-detection will be done as follows:

```
If(bit_cnt==4'd11)
 error<=(parity_error|frame_error)
else
 error<=1'b0;
```

The "bit_cnt" is data bit counter. When the "Start" Bit comes, it begins to count the data frame from "0" to "10" and will return the error flag when bit_cnt==4'd11. So, writes that data frame exactly needs 12 sample clock. The "parity_error" when becomes "1" it means "Odd parity check" error, "frame_error" becomes "1" which means "Stop Bit" check error and "Error" is the overall error flag. When it becomes "1", the online-programming IP will send 00H to PC and the data cannot be written in the SRAM interface management module. And it won't increase the address of the SRAM.

(2) Data Submitting

When the online-programming IP receives a data frame, it will check the data frame receiving format, as it is shown in Fig **5,** and then it will produce a flag as the feedback data according to the check result, as shown in Table **1**. And then it will send FFH or 00H to PC, as shown in Fig. (**6**).

2.5. SRAM Interface Management Module

The SRAM writing time sequence is designed based on RA_512x8 model generated by memory compiler. As shown in Fig. (**7**), it is the SRAM Model Writing Time Sequence. ADD is the address signal, and CEB is the select signal. WEB is a read/write select signal and DATAIN is the data flow [11].

As shown in Fig. (**8**), it is the simulation result. The addr[8:0] is a 9-bit address signal and CEB is the chip select signal (low voltage effective). The WEB is a read/write select signal (read is high and write is low) and data_final[7:0] is an 8-bit data signal.

As shown in Fig. (**2**), if the "Start" signal is "0", MV10 operates in download mode. When the "Start" signal is "1", MV10 operates in normal mode.

When MV10 operates in download mode, the SRAM's select signal CEB and Write Enable signal WEB are effective. The correct data will be written into the SRAM and the SRAM address will increase automatically. If the data frame isn't correct for parity_error or frame_error , it would not be written into the SRAM and the SRAM address remains unchanged.

When MV10 operates in normal mode, SRAM's select signal CEB and Write Enable signal WEB are ineffective. The data frame would not be written into the SRAM, and the program address will change normally.

2.6. State Machine for Erasing and Programming

Before SRAM Interface Management Module can operate normally, the State Machine for Erasing and Programming should be designed successfully. The two state machines can provide the control logic signals to make the SRAM operate well [12]. When the erasing state machine is over, it can automatically jump into the programming state machine.

(1) Erasing State Machine

As shown in Fig. (**9**), the erasing state machine has 8 states. The states 0-3 are used to send eraser order to SRAM, including two clear instructs, one setup instruct, one wakeup instruct and two erasing instructs. And then the state 4, the expected erasing RAM address will be given and then the Read Enable signal is effective. The states 5 and 6 make it read the data successfully. It should be designed to make sure whether the erasing procedure is over or not when it is

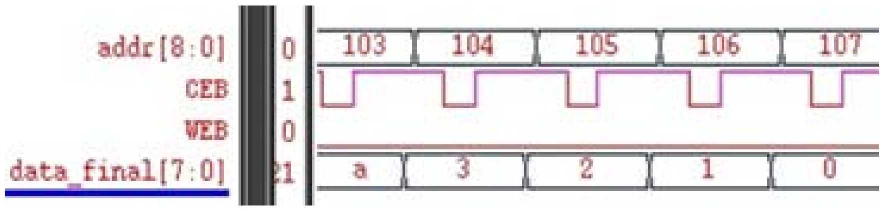


Fig. (8). Designed SRAM Write Sequence.

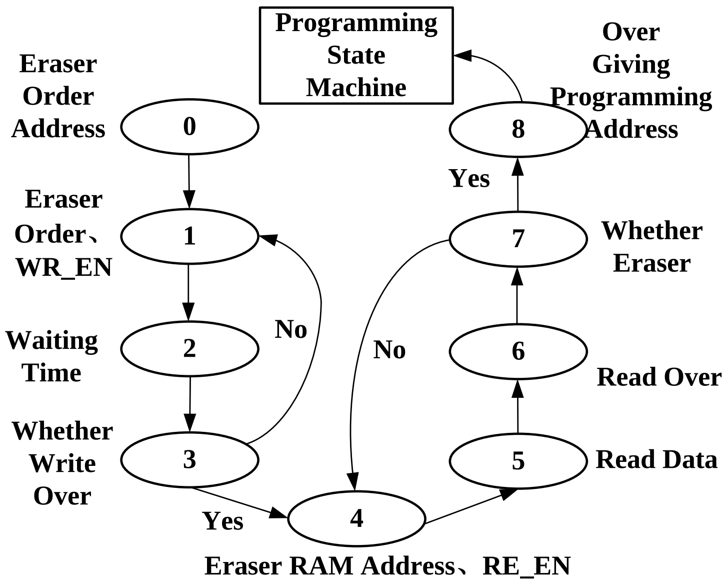


Fig. (9). Erasing State Machine.

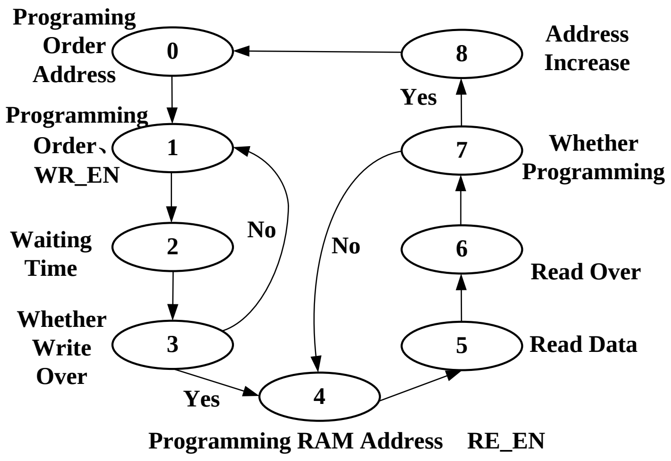


Fig. (10). Programming State Machine.

in state 7. Lastly, when the erasing is over, it should give the programming start address and end address to the address register for the programming state machine.

(2) Programming State Machine

As shown in Fig. (**10**), the programming state machine has 8 states. The states 0-3 are used to send the programming order to SRAM, including two clear instructs, one setup instruct, one wake-up instruct and two programming instructs. And then the state 4, the expected programming RAM address will be given and then the Read Enable signal is effective. The states 5 and 6 can make it read the data successfully. It should be designed to make sure whether the programming procedure is over or not when it is in the state 7. And then when the current address programming is over, it should increase the programming address so that it can operate the next programming or other orders.

The erasing state machine and programming state machine can provide the control logic signals to make the SRAM Interface Management Module operate well.

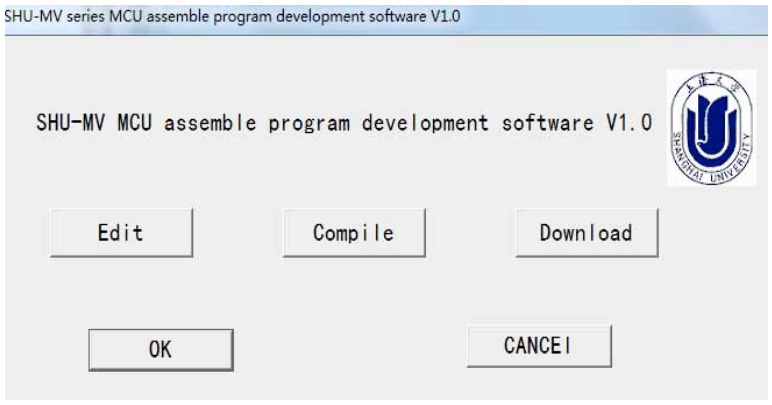

SHU-MV series MCU assemble program development software V1.0
SHU-MV MCU assemble program development software V1.0
Edit
Compile
Download
OK
CANCEl

Fig. (11). MV_IDE Main User Interface.

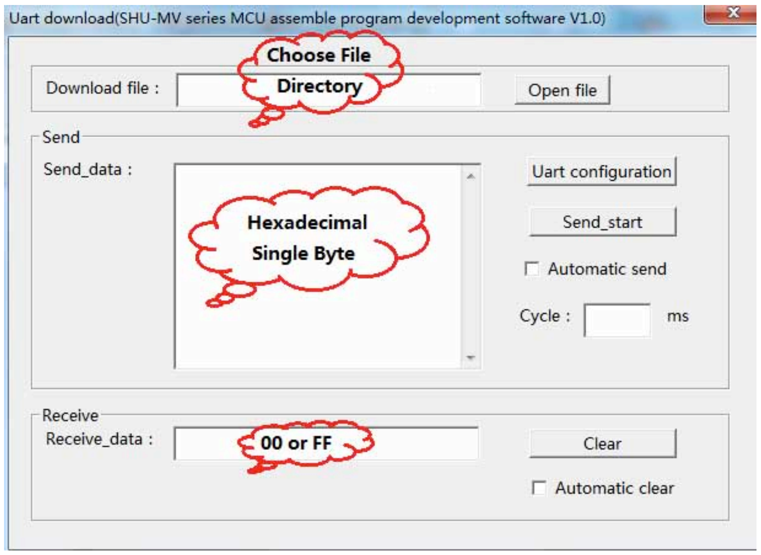
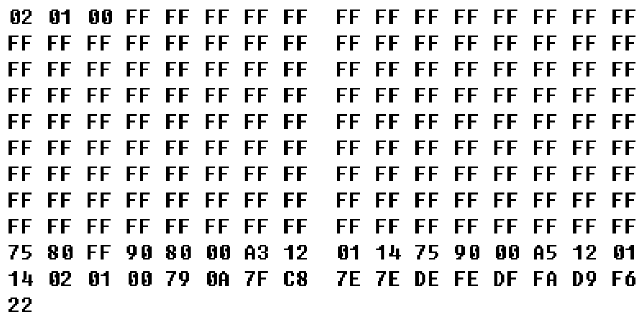


Fig. (12). MV_IDE Program Download Interface.



```
02 01 00 FF FF FF FF FF  FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
75 80 FF 90 80 00 A3 12  01 14 75 90 00 A5 12 01
14 02 01 00 79 0A 7F C8  7E 7E DE FE DF FA D9 F6
22
```

Fig. (13). Hexadecimal Single Byte BIN File.

3. SOFTWARE DESIGN

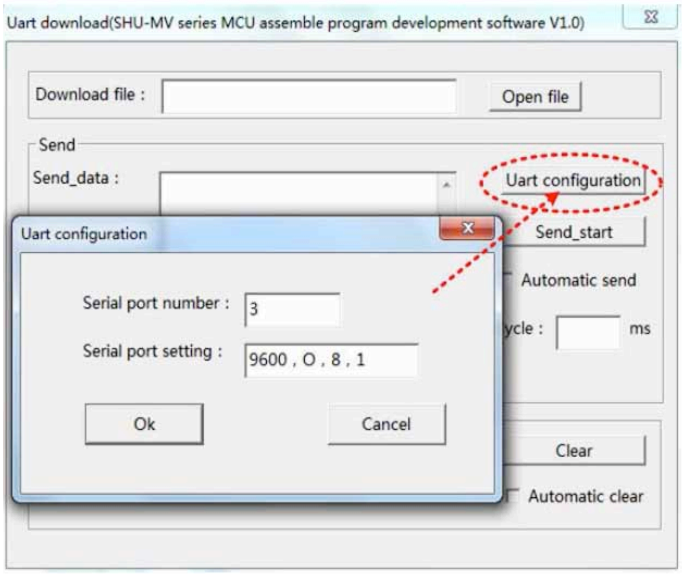
3.1. MV_IDE Overview

As shown in Fig. (**11**), it is the main User Interface(UI) of the Shanghai University Assemble program development software [13]. It's the Assemble program development platform for MV series MCU. Assemble language editor and compiler based on INTEL8051 instructions are integrated into the Integrated Development Environment (IDE). The new function of online-programming in the new generation of MV series MCU which is able to write machine code into the MV10's program SRAM has been realized successfully.

3.2. Program Download Function Design

The program download function can send BIN file compiled by MV_IDE to MV10 through serial port. MV_IDE determines whether to send next data frame or re-send current data frame according to the error flag of the online-programming module.

By clicking the Download button in the main UI, it can open the program download interface, as shown in Fig. (**12**) and then select the BIN file. MV_IDE reads BIN file and converts it into hexadecimal single byte (shown in Fig.**13**). The file will be saved as bin.txt in the project directory so


Uart download(SHU-MV series MCU assemble program development software V1.0)

Download file :

Send

Send_data :

Uart configuration

Serial port number : 3

Serial port setting : 9600 , O , 8 , 1

Fig. (14). Serial Port Configuration Interface.

that it can be send byte by byte. As shown in Fig. (**12**), it is the Uart download(SHU-MV series MCU assemble program development software V1.0). It mainly includes Download file, Sending setting and Receiving setting. The download file directory can be set. When sending data, Uart configuration is used to configure the serial port, and sending mode can be chosen for automatic send or manual operation. And also when receiving data, the data can be cleared automatically or manually.

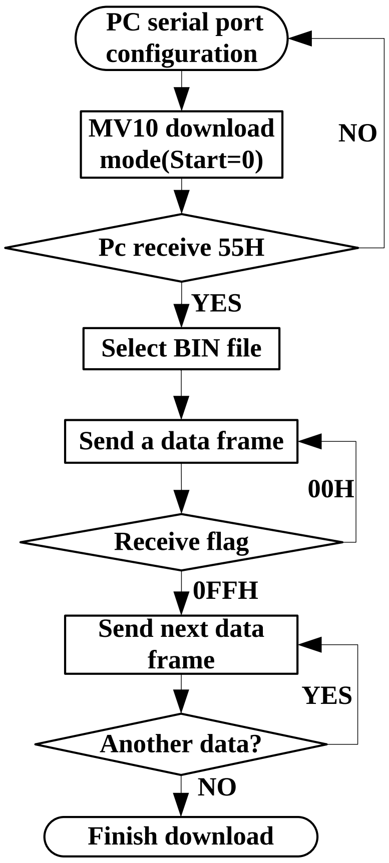


Fig. (15). Program Download Flow.

As is shown in Fig. (**14**), it is the serial port configuration interface. “Serial port number” is used to select the serial port number of PC. The first part of “Serial port setting” is baud rate while the second part is checking bit. N means no checking bit, O is odd checking and E is even checking. The third part is the bit-width of the data and the last part is the bit-width of the stop bit. The data (9600,O,8,1) is used in common situation. It means that the data flow’s baud rate is 9600 Bit/s with odd checking, 8 bit-width, and one bit-width of the stop bit.

Fig. (**15**) is the program download flow. PC serial port should be configured first. And the “Start” signal should be "0", then the MV10 can operate in download mode. And then the data submitting module will send “55H” to PC as a handshaking signal that it is ready to receive data. If the communication is OK, the Bin file selected can be converted into a hexadecimal single byte so that it can be send byte by byte. Then the data receiving module will operate. It will check the data format, and then produce a flag according to the check result. And, also the data submitting module will send FFH or 00H to PC.

4. SIMULATION AND VERIFICATION

4.1. Simulation Based on Modelsim

Fig. (**16**) shows the simulation of writing data into the SRAM [14]. The 8-bit data signal “data_final” is written into SRAM. Data frame “10111010” artificially makes error to test the error detection mechanism. So PC will send this data again after detecting the error flag. SRAM address will increase until the data detection is correct.

4.2. FPGA Verification

As shown in Fig. (**17**) and Fig. (**18**), they show the simulation of writing 16 data frames into the SRAM of MV10 embedded with the online-programming IP by the Uart download(SHU-MV series MCU assemble program development software V1.0). Fig. (**17**) is the PC interface of FPGA Verification and Fig. (**18**) is the FPGA Verification

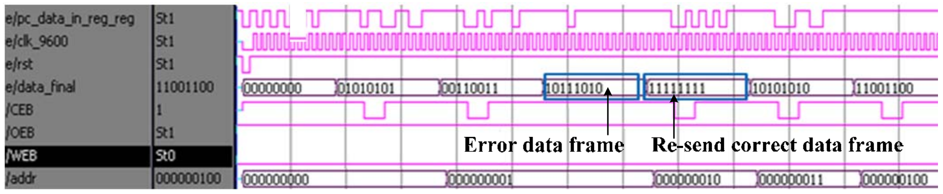
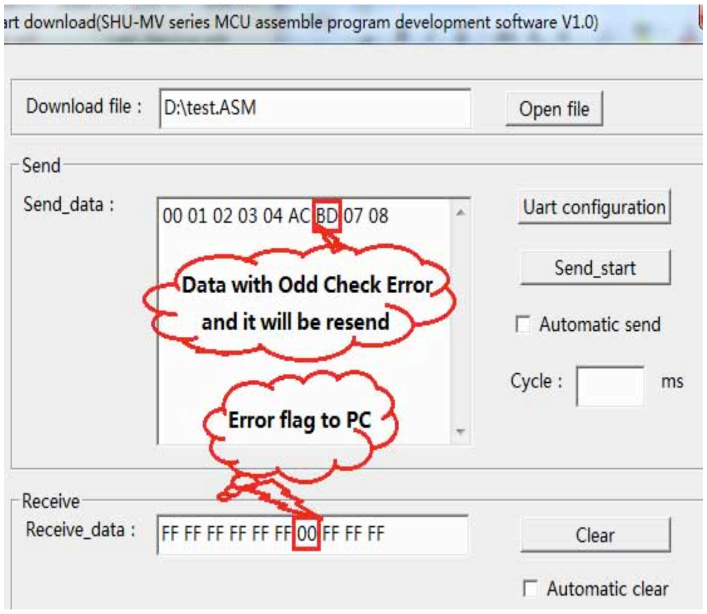


Fig. (16). Modelsim Simulation Result.


rt download(SHU-MV series MCU assemble program development software V1.0)
Open file
Send
Uart configuration
Send_start
Automatic send
Cycle :
ms
Data with Odd Check Error and it will be resend
Error flag to PC
Receive
Clear
Automatic clear

Download file : D:\test.ASM

Send_data : 00 01 02 03 04 AC BD 07 08

Receive_data : FF FF FF FF FF FF 00 FF FF FF

Fig. (17). PC interface for FPGA Verification.

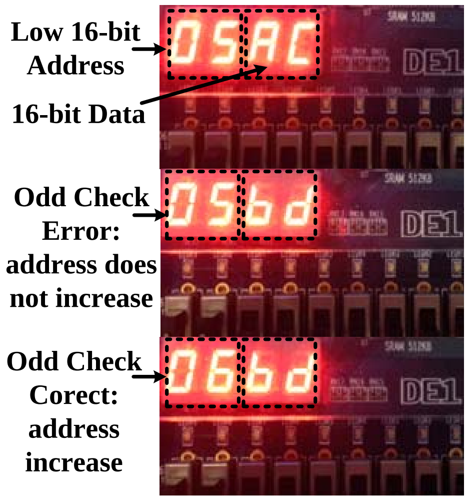

Low 16-bit Address
16-bit Data
Odd Check Error: address does not increase
Odd Check Corect: address increase
05AC
05bd
06bd

Fig. (18). FPGA Verification Test Board.

Test Board. Altera DE1 FPGA board is used as the test board connected to PC through serial port wire. In the FPGA board, left two 7-seg represent the SRAM's low 16-bit address and the right two 7-seg represent the 16-bit data written into the SRAM of MV10. The seventh data frame sets error artificially, and then the error flag "00" is submitted to PC by data submitting module and then the PC will send the same data again, as shown in Fig. (**17**). When the data receiving module receives the correct data, it will be written into the SRAM of MV10 and also the address of the SRAM will increase automatically, as shown in Fig. (**18**).

Through a large amount of experiments with odd check error and frame error randomly, all the results meet our expectation.

4.3. Board Level Verification

Fig. (**19**) is the board level verification based on MV12 [15], which is a multi-core heterogeneous embedded SoC, independently developed by the authors which has full intellectual property rights of the ASIC and has passed the silicon verification with Global Foundry 0.35um mix-signal library. In order to test the System on the Chip with online-

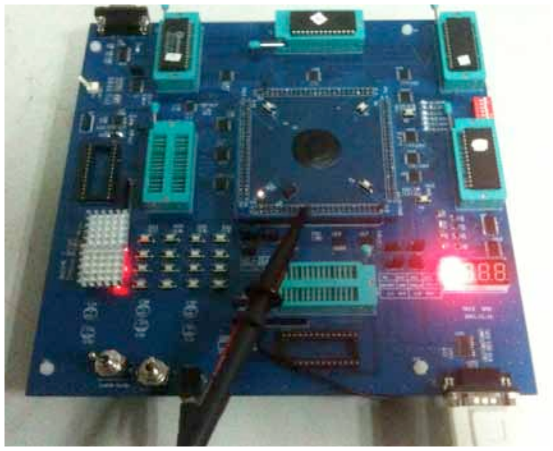


Fig. (19). Board Level Verification.

programming IP core on board level, writing the project program which has already been realized by off-chip EEPROM into the on-chip program SRAM through PC. It can also achieve the corresponding functions above.

5. CONCLUSION

The thesis mainly provides an overview of the design and verification for an online-programming IP core based on a SoC. The online-programming IP core can write BIN file into the on-chip SRAM, which is mainly controlled by erasing state machine and programming state machine, for program debugging with high accuracy-communication and error self-detection. The IP core has passed Modelsim simulation, FPGA verification and board level verification. It has already been used in the development of MV10 SoC program. Although the design is based on MV10, it can also be used in other SoCs.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflicts of interest.

ACKNOWLEDGMENTS

The research work is supported by the National Natural Science Foundation of China (*No. 61376028*) and the Science and Technology Commission of Shanghai Municipality (No. 13111104600).

REFERENCES

[1] Y. Zhaoyang and L. Yongge, “A SCI online programming methodology based on the software”, *Suzhou University Journal*, vol. 30, no. 2, pp. 45-49, 2010.

[2] H. Yue-li and X. Bing, “design of an embedded on-chip debug support module of MCU”, The 8th IEEE CPMT Conference on High Density Micro-system Design and Packaging and Component Failure Analysis(HDP’06), IEEE, pp. 278-281, 2006.

[3] H. Jun-liang, H. Yue-li, and W. Kun, “Interface Design of MV10 MCU based on AHB”, *Computer Measurement & Control*, vol.19, no.12, pp. 3089-3095, 2011.

[4] H. Yue-li, H. Jun-liang, Z. Jun, “Implementation of advanced peripheral bus interface for MV10 MCU”, *Journal of Shanghai University*, vol.15, no.4, pp. 287-291, 2010.

[5] B.M Tudor, “Understanding Off-Chip Memory Contention of Parallel Programs in Multicore Systems”, *2011 International Conference on Parallel Processing (ICPP)*, pp. 602-611, 2011.

[6] S. Banerjee and T. Gupta, “Efficient online RTL debugging methodology for logic emulation systems”, *2012 25th International Conference on VLSI Design*, pp. 298-303, 2012.

[7] H. Yue-li and X. lei, “Reusable Design of CAN Bus Controller IP Core. 4th International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2012”, *Applied Mechanics and Materials*, pp. 255-260, 2012.

[8] G. Lamei and H. Yue-li, “A kind of MCU bus architecture design”, *Computer Measurement & Control*, vol.13, no.7, pp.715-717, 2005.

[9] H. Park, J. Xu, J. Park and J-Hoon Ji, “Design of on-chip debug system for embedded processor”, *IEEE International SoC Design Conference(ISOCC), ISOCC’08*, vol.3, pp.11-12, 2008.

[10] T.P. Blessington, B.B. Murthy, G.V. Ganesh and T.S.R Prasad, “Optimal Implementation of UART-SPI Iinterface in SOC. 2012”, *International Conference on Devices,Circiuts and Sytems, (ICDCS)*, pp. 673-677, 2012.

[11] B-Do Yang, “A low-power SRAM using bit-line charge-recycling for read and write operations”, *IEEE Journal of Solid-State Circuits*, vol.45, no.10, pp. 2173-2183, 2010.

[12] I. García-Vargas, R. Senhadji-Navarro, G. Jiménez-Moreno,A. Civit-Balcells and P. Guerra-Gutiérrez, “ROM-Based finite State Machine Implementation in Low Cost FPGAs”, *IEEE International symposium on Industrial Electronics(ISIE)*, pp. 2343-2347, 2007.

[13] Shanghai university MCU assembly program development platform software（MV_IDE）V1.0 user guide.

[14] H.Yueli, Z. Chen, “Establishment and Verification of an IP- Core Based System on Chip”, *Computer Measurement & Control*, vol.18, no.3, pp. 629-631, 2010.

[15] W. Longjie, H. Yueli, L. Tingyao, Y. Chao, “Design and implmentation of interrup controller based on MV12 Multi-Core SoC”, *Semiconductor Technology*, vol. 37, no.10, pp. 755-759, 2012.

Received: August 13, 2013 Revised: August 28, 2013 Accepted: August 28, 2013