Comparative analysis of methods for digital system debugging at the design stage

A D Ivannikov

Institute for design problems in microelectronics of Russian Academy of Sciences, 3 Sovetskaya st., Moscow, 124365, Russia

E-mail: adi@ippm.ru

Abstract. The paper discusses debugging methods, that is, identifying and correcting errors in designs of digital systems specified by the circuit diagram of hardware and the text of software or firmware. Debugging is considered on a fabricated layout of hardware and on a software model of a digital system, as well as on a combination of the layout of individual blocks and a software model of the rest of the system. The methods of organizing debugging modes for all considered methods are analyzed. It is indicated that for digital systems on a chip, the most effective method is to debug design using a software model.

1. Introduction

When designing digital systems, the result of which is the circuit diagram of the blocks and elements of digital systems, as well as the text of software or firmware, an important stage in the development is debugging, that is, the identification and correction of design errors [1-3]. With the ever-increasing complexity of modern digital systems, the stage of debugging the circuit diagram of hardware, software, and also their joint work can occupy a significant part of the entire design phase [1, 3, 4].

2. Components of the debugging process

For debugging of digital system design, the following components are required.

1. Debugging object, which allows to determine the reaction at the output from a given input action. The debugging object can be either a mock-up or prototype of a digital system, or its software model.

2. A finite set of time-finite input actions, we call them debugging tests, which are fed to the debugging object during debugging experiments. The set of debugging tests must have the property of completeness, that is, from the correctness of the reaction of the debugging object to all tests of the set, the correctness of its operation should follow for any admissible input influences.

3. The criterion for determining the correctness of the output response of the debugging object to the input signals.

4. The ability to localize errors in a digital system if the debugging object responds incorrectly to a given input action.

5. Means for making corrections to the debugging object and repeating the debugging experiment.

Let us analyze the possible methods of debugging digital systems at the design stage, considering alternately the components of the debugging process.
3. The object of debugging and methods of digital system debugging at the design stage

Debugging of a digital system can be carried out both on the fabricated sample of the digital system, and on some computer model of the latter [4-6]. In accordance with the various representations of the developing digital system, four debugging methods can be distinguished that are possible at the design stage, namely:

- a method of full prototyping;
- a method of partial prototyping using a typical core of hardware;
- computer simulation method;
- a computer simulation method using some real blocks of hardware.

With the method of full prototyping, the start of debugging of hardware and software and firmware and their joint work is possible only after the manufacture of the layout, which requires material costs and takes a rather long time.

In the process of analyzing the operation of the digital system and localizing errors in the design of the circuit diagram and firmware, it is necessary to monitor changes in the contents of the system registers and memory cells, as well as logical signals on the buses and lines of hardware, to be able to stop and resume the execution of the program or firmware, set and modify the contents of internal system registers and memory cells. If the tracking of changes in logical signals at the external outputs of the system in the layout can be carried out using logic analyzers, then control over changes in the contents of internal registers and memory cells that do not have external outputs, stop and restart the program or firmware, setting and changing the contents of internal registers and memory cells, that is, debugging modes, are not possible if special tools are not provided inside the layout. These operations can be carried out only if the layout has additional hardware blocks, special software, and the connection of interactive input-output devices, for example, the JTAG system [5, 7], which is often provided for in the developed digital system. In some cases, the digital systems under development include integrated logic analyzers located on the chip and ability to be connected to certain logical nodes using control logic signals. Additional tools built into the digital system greatly facilitate the debugging process, but not to the full extent. In addition, the development of additional hardware and software for the effective organization of debugging modes on the layout is non-trivial.

Debugging a digital system on the layout of hardware at specific values of the dynamic parameters of the blocks does not guarantee the operability of the hardware of the system for any values of dynamic parameters of blocks that are permissible according to the technical specifications.

Debugging the layout together with the managed object in real time is often difficult due to the inability to create real, for example, emergency situations at the object. Such debugging can be carried out by joint modeling on a computer of a digital system and the behavior of a managed object in the same time scale.

The method of partial prototyping involves dividing the hardware of the developed system into a type of core (central part), the same for particular class of digital systems, and part, for example, of the interface circuit with a specific controlled object, present only in the developing system. A typical kernel, including hardware and software for debugging modes, can be made and debugged once, after which it is used in debugging various digital codes of a certain class in combination with a mock-up of a non-standard part.

So, in many of the simplest digital control systems (controllers), the bulk central part (processor unit, memory, pairing with typical external devices, typical input-output channels) is typical, and therefore the partial prototyping method for such systems is applicable to a larger degrees than for digital systems of other classes [6, 8].

In the case of microprogrammable systems, which are characterized by a wide variety of structures and architectures, the microprogram control unit and the memory of high-capacity microprograms are included in a typical core, and the data processing unit, together with the interface circuits with the serviced object, is mocked up in accordance with the circuit diagram of the developing system. Moreover, the prototyped part of the system is significantly larger than in the case of systems based on computers with a fixed system of instructions.
Due to the fixed organization of the microprogram control block of a typical core, the partial prototyping method does not allow obtaining debugging objects for the whole variety of microprogrammable digital systems.

Thus, characterizing the method of partial prototyping as a whole, we can say the following.

1. The method requires less cost for the manufacture of the layout than the method of full prototyping.
2. The method can be effectively used in the design of systems that have a significant amount of type core technology. Application of the method with an atypical structure and architecture of the developed digital system is impossible.
3. The method of partial prototyping is characterized by a number of disadvantages inherent in the prototype method in general, namely:

   • the complexity of making changes to the circuit diagram in the process of debugging;
   • the inability to guarantee the operability of technical equipment for any allowable spread of block delays;
   • the difficulty of debugging the layout of the system together with the serviced object in real time due to the inability to artificially create a number of situations at the object.

The computer simulation method involves the use of the program model of the designed system as a debugging object [1, 5, 9-12]. This method is universal in the sense that the model can be obtained for a digital system of any structure and architecture, including non-standard microprogrammable and multiprocessor systems.

Checking the operability of the technical means of the digital system for all acceptable values of the dynamic parameters of the blocks at the same time is possible only by computer simulation. This method essentially allows debugging by joint modeling of a digital system and a serviced object in all possible operating modes of the latter [8]. Using the modeling method reduces the time and cost of the debugging process due to the lack of the need to make a layout until the end of debugging of the design.

The disadvantage of the modeling method is the slower performance of the program model of the debugging system relative to the layout. So, when simulating at the level of a system of instructions, emulation of one instruction of a processor element of a digital system accounts for several tens to several hundreds of instrumental computer instructions. With functional-logical modeling of the system at the block level, the amount of a machine resource, as a rule, are even greater.

Nevertheless, the use of computer models of digital systems as a debugging object is very effective due to its versatility with respect to the type of debugged systems, the lack of the need to make a layout, and the possibility of parallel analysis of several technical solutions.

The use of computer simulation using real blocks is due to the following. As mentioned above, the disadvantage of using computer models of digital systems is the long simulation time. When modeling at the functional-logical level, usually each block has its own program model, and the hardware model as a whole is generated by a combination of block models in accordance with a given circuit diagram of blocks. Models of digital blocks of ultrahigh degree of integration are difficult to develop, take up a large amount of memory and require a lot of computer time in simulation. In this regard, it is known the use of automated workstations for design, in which functional logic modeling at the block level is used for debugging, and the central computing core is physically connected to the system. In this case, however, two problems arise: the lack of access to the internal registers of the physically connected block during debugging and the significantly different speeds of the physical block and the software model.

Thus, the method of computer simulation using real blocks can reduce the time spent on simulation. Nevertheless, this method does not guarantee the operability of the system being debugged for any acceptable values of the dynamic parameters of the hardware-connected units. In addition, since the speed of the debugging object in this case is determined by the simulating part and significantly less than the speed of the real system, it is difficult to guarantee the system’s operability in real mode of operation.

The simulation method using real blocks should be considered as an extension of the simulation method.
4. Error localization in digital systems designs

The only universal method for localizing errors both in the hardware circuitry and in the firmware at the stage of design debugging is the developer’s analysis of internal variables: the contents of memory cells and registers, as well as logical signals on the buses and lines of the digital system during its operation. Selective and detailed analysis of these internal variables in combination with knowledge of the required functioning of the developing system allows the developer to assume and check for specific errors in the project.

Thus, the possibility of a convenient and detailed analysis of changes in the internal variables of a digital system is the main means of localizing design errors.

When using the full prototyping method, tracking of logical signals on the system buses and block outputs is carried out using various types of logic analyzers, including those built into the designed system. These devices allow you to check the logic diagrams of the nodes of the circuit diagram when the specified conditions occur.

Tracking the change in the contents of internal registers of blocks and cells of the memory, as well as the organization of debugging modes of firmware using the full prototyping method, is carried out using the system of debug monitors loaded into RAM. However, without additional hardware for organizing debugging modes, software execution is slower than in real hardware. Debug monitors focused on tracking contents of memory cells and registers of the standard part do not allow analyzing the status of registers of atypical interface circuits with external devices.

In the case of non-standard microprogrammable and multiprocessor systems, the use of typical debug monitors is not possible. Creating monitors for each particular system is expensive and is a non-trivial task.

The partial prototyping method involves the use of special hardware and software debugging complexes as a typical core, which are universal digital systems with a large amount of memory (including external memory on magnetic media), a wide range of external devices, including logic analyzers, and a resident development system and debugging software. In the debugging complexes, special hardware and software tools for organizing debugging modes are provided, which make it possible to stop at the necessary points of the program or microprogram, monitor changes and modify the contents of memory cells and registers of the typical system core.

Significantly expanding the capabilities of debugging systems is the presence of in-circuit emulators - hardware units that include a processor that is architecturally equivalent to that used in a debugging system, and hardware-implemented circuits for tracking logical signals on the system bus, which are connected to a mockup of hardware of a debugged system instead of a microprocessor. The use of in-circuit emulators allows for the joint debugging of hardware and software in real time, as well as connecting the layout of a digital system to a serviced object for joint debugging. In this case, it is possible to gradually build up debugging hardware by sequentially transferring functions from the typical core boards located in the debugging complex to the added layout boards. In combination with logic analyzers, hardware-software debugging systems are a powerful tool for debugging digital systems.

In the case of digital systems using microprogramming, the debugging complex, as a rule, includes RAM, simulating the ROM of microcommands, as well as a universal computer for controlling the operation of the complex. The presence of a typical microprogram control unit allows you to create system software that provides the organization of debug modes for executing microprograms and tracking logical signals on the system bus, which is especially necessary in the case of debugging microprogrammed systems. However, such complexes can be used for debugging microprogrammable digital systems only with a standard bus structure and a standard bus information protocol, as well as a typical microarchitecture, which limits the use of debugging complexes for microprogrammable systems to typical or similar to typical systems.

In the case of multiprocessor systems, hardware-software debugging systems are also used. A number of well-known hardware-software debugging complexes have in-circuit emulators that allow
you to simultaneously monitor the operation of two or more processor units included in a concentrated multiprocessor system.

Hardware-software debugging systems, used simultaneously as a standard kernel with built-in debugging tools and as an automated workstation for developing software and, in some cases, microprogramming software, is one of the main factors that determine the widespread use of partial prototyping when debugging digital systems.

When using the simulation method for debugging, tracking both changes in the contents of memory cells and internal registers of all system units and the logical values of signals in all nodes of the circuit does not cause difficulties, since they appear to be variables in the models. The convenience of setting any debugging modes, information output conditions, and information about the functioning of the model itself is determined by the organization of the simulating system and the language for controlling its operation.

In recent years, methods for working with big data have found application in digital systems simulating systems, that is, not highlighting the most significant variables when modeling and searching for errors, but remembering when modeling a system at the level of register transfers of all data, regardless of their significance. In this case, when the error is found in any variables: logical signals or contained registers, a reverse look at the simulation progress is possible to detect the cause of the error [1].

When applying the simulation method using real blocks, it is necessary to choose the mentioned blocks as the ones whose tracking of the internal registers is not necessary for debugging.

5. Making corrections to the design
Corrections to the text of software and firmware located in RAM are carried out by conventional methods and do not cause difficulties. Making changes to the circuit of hardware with the method of full or partial prototyping requires a physical change in the circuit: changes in connecting blocks to circuit nodes, deleting, adding or modifying blocks. This requires a certain amount of time, especially if there are significant changes in the circuit. Changes to the simulated circuit of hardware are made by editing a computer model, which is carried out by the developer in an interactive mode and practically does not require time-consuming.

6. Debugging tests and determining the correctness of the output reaction
Choosing a set of debugging tests and determining the correctness of the output reaction is a common task for all methods of debugging digital systems.

The task of choosing debugging tests differs significantly from the task of choosing checking and diagnostic tests of digital equipment used at the production and maintenance stage, in that there are no models of possible errors when debugging digital systems at the design stage. The problem of choosing debugging tests for both hardware and digital systems as a whole is somewhat similar in its formulation to the problem of synthesizing test cases for debugging programs. Debugging tests should be selected primarily on the basis of the functions performed by the digital system and determined by the terms of reference for its development, as well as, possibly, the structure of the developed system [12-14].

Regardless of the method of selecting tests during debugging, it is necessary to determine the correctness or erroneous reaction of the debugging object. The most convenient, but not always possible in design practice is the case when the developer knows exactly the required reaction of the debugging object to any test. Often, the developer in advance does not exactly know the correct reaction to the debugging test, but after receiving the result and, possibly, analyzing the intermediate data, he can tell whether the received reaction of the debugging object matches the required one. In any case, the ability of the developer to determine the correct response of the debugging object to the input is a necessary condition for debugging in general.
7. Conclusion

Thus, based on an analysis of the methods of debugging digital systems (table 1), it can be said that potentially only a computer simulation method is universal with respect to the structure and architecture of debugged digital systems. This method is most widely used in the development of modern digital systems. However, other methods also find application.

Table 1. Comparative analysis of debugging methods for digital systems designs.

| Indicator                              | Full mock-up method | Partial mock-up method | Simulation method | Simulation method using physical blocks |
|----------------------------------------|---------------------|------------------------|-------------------|-----------------------------------------|
| Class of debugging systems             | Any                 | Systems having a typical core | Any              | Systems having selected blocks          |
| Organization of debugging modes        | Difficult           | Simple, excluding tracking of registers of atypical part | Simple          | Simple, excluding tracking of registers of physical blocks |
| The complexity of making changes to the circuit diagram | High               | High                    | Low              | Low                                     |
| Ability to debug a design before making a layout | No                 | After making a layout of atypical blocks | Yes             | Yes                                     |
| Cost                                   | High                | Medium                  | Low              | Low                                     |
| Duration, including mock-up time, if necessary | Large              | Medium                  | Small            | Small                                   |

References

[1] Karnane K and Goss C 2015 Automated Root-Cause Analysis to Reduce Time to Find Bugs by Up to 50% https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/system-design-verification/indago-debug-platform-wp.pdf last accessed 2020/03/21

[2] Vishnekov A V and Ivanova E M 2019 Simulation of the Super-scalar Processor Core Operation J. Phys. Conf. Ser. 1163 012010

[3] Stempkovsky A L and Ivannikov A D 2019 Russian Microelectronics 48(5) 318-25

[4] Alexopoulos I 2012 How to Debug Embedded Systems https://www.edn.com/how-to-debug-embedded-systems last accessed 2020/03/21

[5] Garashchenko A V, Putrya F M, Gagarina L and Dzhurakulov A A 2019 Proc. IEEE Conf. of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) (Saint Petersburg and Moscow, Russia: IEEE) pp 1876-9

[6] Považan I, Kmrjetin M and Četić N 2015 23rd Telecommunications Forum, TELFOR (Belgrade, Serbia: IEEE) pp 1005-8

[7] Hu X, Jin Y and Li Z 2018 Proceedings of 2018 IEEE Int. Conf. of Safety Produce Informatization, IICSPI (Chongqing, China: IEEE) pp 527-30

[8] Klinachev N V and Shaburov P O 2017 Proc. 2017 Int. Conf. on Industrial Engineering, Applications and Manufacturing, ICEIAM (Chelyabinsk, Russian Federation: IEEE) pp 1-5

[9] Lin Y and Su A W Y 2011 Proc. 24th IEEE Int. System on Chip Conf., SOCC (Taipei, Taiwan: IEEE) pp 201-6

[10] Shi J, Liu W, Jiang M, Che H and Chen L 2013 Proc. IEEE Int. Conf. on Intelligent Rail Transportation, ICIRT (Beijing, China: IEEE) pp 71-4

[11] Nguyen M D and Kunz W 2012 Proc. 4th International Conference on Communications and Electronics, ICCE (Hue, Vietnam: IEEE) pp 465-70
[12] Ivannikov A, Pozdneev B, Romanova I and Tumkovskiy S 2018 Proc. Moscow Workshop on Electronic and Networking Technologies, MWENT (Moscow, Russian Federation: IEEE) 1-5

[13] Kamkin A, Lebedev M and Smolov S 2016 Proc. 2016 IEEE East-West Design and Test Symp., EWDTS (Yerevan, Armenia: IEEE) pp 1-4

[14] Ivannikov A and Pozdneev B 2019 Proc. Int. Russian Automation Conf., RusAutoCon (Sochi, Russian Federation: IEEE) pp 1-5