Comparison of Two Designs for the Multifunction Vehicle Bus

In this paper, two designs for the decoder of the multifunction vehicle bus (MVB) are compared. The first one follows a bottom-up methodology and the second one has been created in a top-down style. Although this latter methodology is more systematic and easy to automate, it results in a lower performance. In the case of the MVB decoder, the ratio of bottom-up performance to the top-down one ranges from 1.90 to 4.12, depending on the synthesis tool and the device. Selecting as reference the tool and field-programmable gate array (FPGA) that use the fewest logical elements, the bottom-up design can work 2.3 times faster than the top-down one, after two and three iterations for the physical implementation, respectively. In both cases, the circuit has been synthesized on a Virtex- E XCV3200E of Xilinx by Xilinx Synthesis Tool (XST), so that there has been no shortage of physical resources. Therefore, for a particular pair of synthesis tool and device, the final implementation is determined by the design style and not by a hard placement and routing in a hostile fabric. After synthesis, the top-down design was 23.37% larger than the bottom-up design, so the results are not as poor as expected from a nonstructured design; however, this percentage, which is always positive, depends very strongly upon the particular synthesis tool and FPGA. In addition, both descriptions have been completely implemented in a similar CPU time (even the top-down one slightly more quickly, at the first attempt). So the top-down design style is a good candidate to produce circuits in a short time to market (in this case 28% lower), although synthesis tools must be improved in order to increase the performance.

Comparison of Two Designs for the Multifunction Vehicle Bus
Jaime Jiménez, José L. Martín, Member, IEEE, Aitzol Zuloaga, Unai Bidarte, and Jagoba Arias

Abstract—In this paper, two designs for the decoder of the multifunction vehicle bus (MVB) are compared. The first one follows a bottom-up methodology and the second one has been created in a top-down style. Although this latter methodology is more systematic and easy to automate, it results in a lower performance. In the case of the MVB decoder, the ratio of bottom-up performance to the top-down one ranges from 1.90 to 4.12, depending on the synthesis tool and the device. Selecting as reference the tool and field-programmable gate array (FPGA) that use the fewest logical elements, the bottom-up design can work 2.3 times faster than the top-down one, after two and three iterations for the physical implementation, respectively. In both cases, the circuit has been synthesized on a Virtex-E XCV3200E of Xilinx by Xilinx Synthesis Tool (XST), so that there has been no shortage of physical resources. Therefore, for a particular pair of synthesis tool and device, the final implementation is determined by the design style and not by a hard placement and routing in a hostile fabric. After synthesis, the top-down design was 23.57% larger than the bottom-up design, so the results are not as poor as expected from a nonstructured design; however, this percentage, which is always positive, depends very strongly upon the particular synthesis tool and FPGA. In addition, both descriptions have been completely implemented in a similar CPU time (even the top-down one slightly more quickly, at the first attempt). So the top-down design style is a good candidate to produce circuits in a short time to market (in this case 28% lower), although synthesis tools must be improved in order to increase the performance.

Index Terms—Design methodology, logic design, rail transportation electronics, train communication network (TCN)

I. INTRODUCTION

A DEEP revolution has been seen in electronic design during the last decades. The greatest change can be found in digital circuits: from strategies based on standard off-the-shelf components to programmable and application-specific circuits [programmable logic devices (PLDs), field-programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs)].

In order to design a complex digital system, the main philosophies are [1]:
1) bottom-up methodology;
2) top-down methodology.

Manuscript received November 26, 2004; revised February 15, 2005. This paper was supported by Ministerio de Ciencia y Tecnología of Spain and Fondo Europeo de Desarrollo Regional (FEDER) in the framework of the TIC2001-0062 Research Project. This paper was recommended by Associate Editor R. Camposano.

The authors are with the Applied Electronic Research Team (APERT), Department of Electronics and Telecommunications, Escuela Técnica Superior de Ingeniería de Bilbao, University of the Basque Country, 48013 Bilbao, Spain (e-mail: jaime.jimenez@ehu.es).

Digital Object Identifier 10.1109/TCAD.2005.855925

Fig. 1. Design flow for an electronic system.

The electronic-design development of a digital system has always suggested the idea of a progressive refinement from the initial conception to the final implementation. Hence, design flow is divided in consecutive steps in order to establish sequential goals. Each of them involves producing a more and more detailed description of the system or model. This methodology is known as top down. Therefore, some different models describing the same definitive system are created and verified before the final prototype is produced (see Fig. 1). First, the behavioral description is obtained. In this step, structural and architectural features are not of concern and the task is to specify the system’s behavior. The second model is the register transfer level (RTL) one, and the third, the gate netlist (synthesis) is generated [2], [3]. It must be verified whether each of them matches the initial specifications and a description cannot be generated until the previous one has been approved. Creating the following description involves changing the abstraction level to a more refined one; in other words, first models are very close to the system’s behavior and do not care about physical details. On the other hand, the designer may produce some later descriptions that consider floorplanning, placement, and routing. The technology mapping in this methodology consists of expanding the system definition, whereas logic optimization comes in the end.

This methodology is systematic, so it is easier to automate than the other one. Since all the steps are well established and each of them consists of coding in a description language with
rigid syntax and semantics, it is a very suitable method to be performed by a computer.

However, the real design flow has always been very limited by the state of the tools used for electronic design. These usually allowed the designer to start from low complexity levels, such as physical or electrical ones, to higher ones such as functional. These limitations have forced designers to use the bottom-up methodology. This one assumes that predesigned components are available in a database that stores their relevant characteristics. The main disadvantage of this methodology is that the system can be validated by simulation only when all the decisions about the architecture and technology of the final implementation have been taken [4]. Hence, fixing errors and bugs means redesigns, time to market gets longer, and project costs rise, contrary to management’s basic rules.

The new programmable logic devices and the development of simulation and synthesis programs based on hardware description languages, such as very high speed integrated circuit (VHSIC) hardware description language (VHDL), are simulating the use of new design methods that allow a top-down methodology to be applied [5]–[7]. Some design teams have used standard C programming language as an efficient way to describe the highest level model [8]–[11]. This has been the second approach followed by the authors.

The remainder of this paper is organized as follows. Section II describes the train communication network (TCN) and gives details about the multifunction vehicle bus (MVB). Section III explains how the bottom-up design was implemented and which features were obtained. The top-down one is presented in Section IV, including some details about the models used. In Section V, the test cases used in simulation are collected, in order to justify the validation process for the designs. Section VI is concerned with the comparison of both designs, similarities and differences in consumed resources and performance. Finally, some conclusions are presented in Section VII.

II. THE MVB

Communication networks bring important profits to public transport, specifically to railways. As train manufacturers have realized how complex the device interconnection on board is, they have asked equipment suppliers for easier networking resources and to use a common standard. Particularly, an on-board train communications system has been widely demanded for modern railways, so device interoperability, distributed control architectures, and integration with other external networks are made easier by interconnecting all the electronic subsystems [12], [13]. In 1989, the International Electrotechnical Commission (IEC) in collaboration with the Union Internationale des Chemins de Fer (UIC) started a new standardization program with the aim of defining a complete network specification. The definitive version was approved as TCN [14].

The general architecture of TCN includes two bus types (Fig. 2):

1) MVB, which is used for attaching the electronic equipment inside a train vehicle;
2) wired train bus (WTB), which is used for interconnecting the different vehicles of a train.

Two of the most important requirements of this network are high reliability and real-time communication. For this purpose, both bus types have a master–slave architecture to control access to the network. So each vehicle bus and train bus has one master node and several slave ones. This type of architecture ensures time-critical data to be transferred in real time, which is an important requirement in traction control.

In a master–slave architecture, the master is a device that spontaneously sends information, a master_frame, to a number of slave devices. It may give a slave the permission to transmit one slave_frame only within a limited time. The slave is a device that receives information from the bus or sends information through it in response to a request from the master.

The MVB is the industrial data network in charge of interconnecting several devices in a local environment (up to 200 m). Among these devices, there is a master, which is the bus administrator, and multiple slaves, which perform different services at the master command. Among the MVB characteristics, these must be highlighted [14], [15]: physical redundancy, signaling (frame start and end delimitation flags), and frame size (of the useful information): 16, 32, 64, 128, or 256 bits. In all MVB media, signaling speed shall be 1.5 Mb/s ± 0.01%, using Manchester encoding (bit rate = 1.5 MHz or 1.5 Mb/s, bit time = 666.7 ns). Individual data bits, “1” and “0,” in the Manchester system shall be encoded as follows: a “1” by a high level during the first half of a bit cell followed by a low level during the second half; and a “0” shall be encoded by a low
The MVB characteristics involve using a module especially dedicated to the access control of the slave devices to the bus. The tasks that must be performed can be divided into two groups: bit-level tasks (imposed by the standard that regulates the MVB) and word (16 bits)-level tasks:

1) bit-level tasks:
   a) Manchester encoding and decoding;
   b) serial-to-parallel and parallel-to-serial conversion;
   c) error detection;
   d) time control.

2) word-level tasks:
   a) transmission of answers to the master node commands;
   b) data reading or writing for transmission through the bus;
   c) received data processing;
   d) obtaining the data to be sent.

All these tasks must be carried out efficiently and at high speed. In addition, the received bits must be synchronized and the signal must be oversampled to verify that the total length of each bit is within the allowed range. The remainder of the paper explains how the decoder of the receiver has been designed in two different ways.

III. THE BOTTOM-UP DESIGN

Fig. 5 shows the general block diagram of the bus controller for a class-1 MVB device [16], [17]. This circuit is in charge of attending the MVB bus in order to send the information received from a central processor or vice versa. This bus controller performs basically the functions of the data link layer of the MVB. It is in charge of decoding the input signal of the serial MVB and forwards it as the information received in a 16-bit parallel bus. This one is connected with a central processor. In the same way, when the central processor needs to send some data through the MVB, the information in words of 16 bits is transmitted by the 16-bit bus to this system and it sends the information using the MVB protocol. In addition, this circuit indicates to the central processor different events detected in the MVB such as master or slave frame detection, CS errors, Manchester code violation, etc. The circuit has been divided into seven blocks.

The four lower blocks in the figure are used to transform the serial data coming from the MVB to parallel data sent to the central processor. Among them, there is the decoder that has been created in a bottom-up way.

The block diagram of the decoder can be seen in Fig. 6. It waits until a correct SB is detected. After that, it identifies the start_delimiter as slave or master. Next, the frame data must be received. Since the signal of MVB is in Manchester code, it must be decoded first [18]. After that, the data are grouped in 16-bit packets by means of a serial/parallel converter and stored in a 16-bit register. The length of the data sequence is 16 bits in the case of a master frame and, for a slave one, it is determined by the F_code received during the latest master frame. In addition, the decoder checks whether the received signal complies with the MVB protocol, which means that the start_delimiter, frame_data, CS, and ED are received correctly.
The CS verification is done in the CS tester block, which compares the received CS with the calculated one.

Finally, the central processor interface block is in charge of connecting the data bus of the last two blocks, input data bus for the encoder, and output data bus for the decoder, with the bidirectional data bus of the central processor. In addition, the other signals connected between the bus controller and the central processor are conditioned in this block.

In the bottom-up design, some blocks are combined in order to create a higher level module. On the other hand, they have been generated from simpler components by composition and hierarchy. The decoder itself makes up the first level, which is composed of seven submodules (the second level). Four of these have been split in new blocks of the third level, 17 altogether, and so on.

Some of the lowest level components have been taken straightforward from libraries: four standard counters, two shift registers, one first-in first-out (FIFO) memory, and one conventional flip-flop. On the other hand, the others have been created from scratch in VHDL: two finite-state machines, eight specific combinational circuits, and five sequential ones.

**IV. THE TOP-DOWN DESIGN**

On the other hand, a completely new design has been created, following the top-down style [19]. From the specifications document, an algorithmic model has been written. Hence, in this step, electronic issues, such as basic components, connections, bus widths, or delays, were not taken cared of and most effort was focused on the system’s behavior. The first algorithmic description, written in natural language, has been coded in C to produce the first executable model of the system. After simulating it exhaustively (see Section V), the C code has been translated into VHDL, just in a behavioral level. Simulation and analysis processes have been repeated until the observed behavior matches the specifications. Finally, this second model has been validated and translated again into a VHDL description for synthesis.

**A. Differences From Bottom-Up Methodology**

Unlike the bottom-up design flow, this second method does not care about system architecture or something like block structure. As soon as the algorithmic model can be translated into a functional model, the first one is forgotten until synthesis demands new refinements. When the functional model can be compiled, it becomes the definitive model. Moreover, general system behavior has been validated first. Subsequent descriptions are also verified as a whole. The designer does not care about which circuit the system will be implemented in, how it will work, which interconnections will be generated, and some other similar details that are critical in bottom-up designs.
B. The Design of the Decoder

First of all, a flow diagram was used to represent the device requirements. This one depicted its behavior and it turned into the start point for translation to C language. That C code has been the initial model and it was simulated until validation (Fig. 7).

A set of functions and subroutines were added so that reading and understanding the description was easier. Some specific functions are called in order to perform well-defined algorithms: decode data bits from received symbols (read_symbol), exclusive or logic operation (XOR), calculate CRC check sequence (CRC), and frame-sending process (send_bits), above all. Some other functions, “invers,” “parity,” and “exp” perform simple mathematical operations.

The core of the C model was written as a “switch” mode control sentence. This resembles directly the style of a finite-state machine, (Fig. 7), so translation into VHDL can be straightforward [20].

In this way, a VHDL model for simulation was written and successfully used in a virtual MVB bus that allows the verification of circuits synthesized for MVB [21], [22]. This first VHDL description (a second layer in the top-down design flow) cannot be synthesized yet. Dependence on the medium, timing details, and resources to control transitions among the states are some differences from the C model.

Finally, the functional description has been rewritten in order to be synthesized using the same software design tool for a commercial technology (Xilinx) as in the bottom-up case. This has required new refinements, until the VHDL code has been fitted successfully in a Virtex-E XCV3200E. For example, the number of logical states increases dramatically because of the high sampling frequency required for reception (see Fig. 8).

Such a large FPGA has been used in order to have no shortage of physical resources. In this way, the implementation has been determined by the design style instead of a hard placement and routing in an exhausted array of few logical cells. In addition, this device allows the embedding of a microprocessor and some other hardware modules in order to produce a high-class device.

The decoder has been arranged in the following processes: the main finite-state machine, a data register, and the cyclic redundancy check (CRC) generator. The most complex one is the main finite-state machine: 515 lines in VHDL to code 46 states and 122 possible transitions. It is in charge of decoding the input samples from the MVB bus in order to determine whether a valid frame has been received and to interpret the data bits. On the other hand, the data register is made up of 288 D flip-flops, since that is the maximum number of bits to be stored from a received slave frame. The CRC block generates an 8-bit pipelined code from each word of 64, 32, or 16 bits.

V. TEST CASES FOR VALIDATION

Verifying an electronic design is always a critical task [23], especially in railways, where security is very important. Therefore, both circuits have been simulated in an exhaustive set of test cases. As it is suitable to such a decoder, these have been timing proofs, about the relative locations of edges inside a frame and about delays between two consecutive frames. The device has received master and slave frames, some of them perfect, some with acceptable distortion, and other inadmissible ones. Two hundred twenty-two cases have constituted the main set for tests, among them:

1) frames with acceptable, too long, and too short SBs;
2) frames with acceptable and too short end_delimeters;
3) frames with acceptable, too long, and too short data bits;
4) frames with inadmissible shifted edges;
5) too long and too short (in bits) frames;
6) wrong CRCs;
7) a too widely spaced pair of master–slave frames (since, in addition to high reliability, this network requires real-time communication).

These test cases have been performed in a commercial simulator, ModelSim from Mentor Graphics (Fig. 9).

VI. COMPARING BOTH IMPLEMENTATIONS

Both descriptions have been synthesized by three different tools, XST of Xilinx, Leonardo Spectrum of Mentor Graphics, and Quartus II of Altera, on plenty of Xilinx and Altera devices, as Table I shows.

The ratio of bottom-up performance to the top-down one ranges from 1.90 to 4.12 (to 2.94 in the case of Xilinx devices). Therefore, the improvement rates are similar enough to say that they are due to the design style.

On the other hand, the number of logical elements used to implement the designs ranges very widely, depending strongly upon the particular synthesis tool and FPGA. The most
Fig. 8. Details of the VHDL model for synthesis.

Fig. 9. View of one simulation of the virtual network.

homogeneous values have been obtained by XST. The tool and FPGA that use the fewest logical elements, a bottom-up design, of course, have been taken as reference in order to compare more details about both designs.

A. Results About the Implementation of the Bottom-Up Design

The synthesis tool, XST of Xilinx, has mapped the bottom-up description in 184 logical cells or slices. The placement and routing was completed in 25 min 4 s at the first attempt.
by a Pentium 800 CPU. The minimum clock period obtained initially was 10.867 ns and after two iterations, reducing this constraint, it was decreased to 7.693 ns, so the final performance is 130 MHz. The required working frequency was just 24 MHz.

This design was developed in about 250 h, including the time used to carry out the simulations.

### B. Results About the Implementation of the Top-Down Design

The system described in a top-down style has been synthesized in 227 slices; this number is quite similar to the one in the bottom-up design, so the decoder complexity is around 190 slices in the best case. Greater difference appears in the number of routed connections, 1559 against 1033. Nevertheless, the complete description of the circuit has been routed at the first attempt in 19 min 52 s, less than the previous one, with a minimum clock period of 21.580 ns. After three iterations, it has been reduced to 17.664 ns, that is, a performance of 56.612 MHz, so the target speed of 24 MHz has been reached again.

This design was carried out in about 180 h, also including the time used to carry out the simulations, which means a time saved of 70 h, that is, 28% of the time necessary for the bottom-up design.

### C. Comparison of Both Approaches

Anyway, the bottom-up design can work 2.3 times more rapidly than the top-down one. Some details about both implementations can be seen in Table II.

### TABLE II

<table>
<thead>
<tr>
<th></th>
<th>Bottom-up</th>
<th>Top-down</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slices consumed</td>
<td>184</td>
<td>227</td>
</tr>
<tr>
<td>Connections routed</td>
<td>1033</td>
<td>1559</td>
</tr>
<tr>
<td>Iterations</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Performance (MHz)</td>
<td>130</td>
<td>56.6</td>
</tr>
<tr>
<td>CPU time for routing</td>
<td>25 min 4 s</td>
<td>19 min 52 s</td>
</tr>
</tbody>
</table>

The results obtained using the two different methodologies show the intrinsic differences between both of them. In the bottom-up approach, the design was performed using an electronic-circuit-oriented strategy: Each block is designed, verified, and synthesized before it is ready to be integrated in another block at a higher hierarchy level. This means that all blocks have been implemented as finite-state machines or combinational blocks, which are just what a synthesis tool expects as a realizable (and optimal) input for synthesis, place, and route. As a result, the circuit is smaller (it needs less
slices to implement the circuit), faster (it can run at higher frequencies), and simpler (it has less connections among slices). The synthesis tool needs less iterations to get a satisfactory circuit, but it needs more time to perform its task.

On the other hand, in a top-down methodology, the designer starts with a nonsynthesizable approach, which is progressively refined to fix those blocks that cannot be understood by the synthesis tools. The design process stops when the tools are able to perform their task. As the abstraction level used in this approach is higher, the designer can describe and verify the circuit faster than in the bottom-up methodology, but the resulting circuit is not described in a simple “electronic” way that can be easily understood by the synthesis tool. On the contrary, it has the maximum level of abstraction the tool can bear, which produces a larger circuit (more slices to perform the same task) that can only run at a lower frequency and whose connectivity level is much higher. The synthesis tool needs more iterations to find a solution but, as the number of slices and connections is larger, it needs less time to route all signals in the design.

These results suggest that, as the capacity of FPGAs and programmable devices increases, the designer may use a higher abstraction level to describe the circuits, making sure the target device is much larger than the minimum capacity needed for the implemented circuit. This, of course, fits the industry requirements of these days, which allow the design of a board with the largest available FPGA at a reasonable cost, which exceeds the initial needs of the application, but allows the design at a higher abstraction levels, which reduces the time to market. On the other hand, as the hardware has already been designed, verified, and certified, there is no need of repeating this long and costly process whenever a new project must be executed, easing the faster development of new FPGA-based designs.

VII. CONCLUSION

Two designs for the decoder of the MVB have been compared. The first one follows a bottom-up methodology and the second one has been created in a top-down style. When synthesized by different tools on plenty of devices, the bottom-up design provides higher performance in fewer logical elements. Nevertheless, the top-down one has been created in 28% shorter time.

In order to compare some other issues of both designs, the tool and FPGA that use the fewest logical elements has been taken as reference. The top-down design is just 23.37% more complex than the bottom-up one, if the number of logical elements in the FPGA are taken into account, which is not as poor as expected from a nonstructured design. However, this percentage, which is always positive, depends very strongly upon the particular synthesis tool and FPGA.

On the other hand, the synthesis tool has achieved a denser mapped floorplanning in the case of the top-down design. In addition, both descriptions have been completely implemented in a similar CPU time (even the top-down one slightly more quickly, at the first attempt).

However, the bottom-up design can work 2.3 times more rapidly than the top-down one, after two and three iterations for the physical implementation, respectively. In both cases, the circuit has been synthesized on a Virtex-E XCV3200E of Xilinx. Therefore, the top-down design style is a good candidate to produce circuits in a short time to market, although synthesis tools must be improved in order to increase the performance.

C language has shown weaknesses when used to describe more complex circuits. Therefore, a class-4 device, an MV8 bus administrator, is being coded in SystemC [24]–[27] in order to synthesize the system on a chip.

ACKNOWLEDGMENT

The authors gratefully appreciate the suggestions made by anonymous reviewers.

REFERENCES

Jaime Jiménez received the M.S. degree in telecommunications engineering from the University of the Basque Country, Bilbao, Spain, in 1991. He is currently pursuing the Ph.D. degree in telecommunications at the University of the Basque Country.

From 1991 to 1997, he was with the Robotiker Technological Center and the Basque Government. Since 1998, he has been an Assistant Professor in Electronic Technology at the Electronics and Telecommunications Department of the University of the Basque Country. His main research interests are design methodologies, system-on-chip, and digital communications circuits.

Jagoba Arias received the M.S. degree in telecommunications engineering from the University of the Basque Country, Bilbao, Spain, in 2001. He is currently pursuing the Ph.D. degree in telecommunications at the University of the Basque Country.

Since 2001, he has been an Assistant Professor in Electronic Technology at the Electronics and Telecommunications Department of the University of the Basque Country. His main research interests are digital communications circuits and wireless systems.

José L. Martín (M’98) received the M.S. and Ph.D. degrees in electrical engineering from the University of the Basque Country, Bilbao, Spain, in 1985 and 1992, respectively.

From 1989 to 1995, he was an Assistant Professor in Electronic Technology at the Electronics and Telecommunications Department of the University of the Basque Country. In 1995, he became an Associate Professor. From 1995 to 2001, he has been Head of the Electronics and Telecommunications Department. From 2001 to 2005, he has been Vice-Dean of the Faculty of Engineering in Bilbao, Spain. His main research interests are image processing, system-on-chip, and digital communications circuits.

Aitzol Zuloaga received the B.S. degree in electrical engineering and M.S. degree in project management from Simón Bolívar University, Caracas, Venezuela, in 1985 and 1992, respectively, and the Ph.D. degree in telecommunications engineering from the University of the Basque Country, Spain, in 2001.

From 1985 to 1995, he was with different R&D departments of private companies in Venezuela. In 1995, he joined the University of the Basque Country with a predoctoral grant. In 2000, he became an Assistant Professor in Electronic Technology at the Electronics and Telecommunications Department of the University of the Basque Country. His main research interests are image processing, system-on-chip, and digital communications circuits.

Unai Bidarte received the M.S. and Ph.D. degrees in telecommunications engineering from the University of the Basque Country, Bilbao, Spain, in 1996 and 2004, respectively.

From 1996 to 1999, he joined the University of the Basque Country with a predoctoral grant. Since 1999, he has been an Assistant Professor in Electronic Technology at the Electronics and Telecommunications Department of the University of the Basque Country. His main research interests are reconfigurable circuits and system-on-chip.