Wrapper Design for a CDMA Bus in SOC

T. Nikolić and M. Stojčev
Faculty of Electronic Engineering
University of Niš
Niš, Serbia

Z. Stamenković
System Design
IHP GmbH
Frankfurt (Oder), Germany

Abstract — The research conducted in this paper is aimed at developing a CDMA shared bus as the efficient communication architecture for SOC. The main benefits of using this technique relate to reduction of the number of wires on system bus which varies from 25% up to 81%, while the main disadvantage is increase of the latency of processor read and write operations. The structure of a CDMA wrapper as an interface logic between the shared bus and IP connecting to it is described. VHDL models of two wrapper types (master and slave) are developed and verified. Four different implementations of the CDMA coding technique are presented and realized in FPGA and ASIC technologies.

Keywords - CDMA; wrapper; bus; SOC

I. INTRODUCTION

As the number of processing elements increases, the interconnection plays an increasingly major role in system-on-chip (SOC) design. The type of interconnection used for a specific application will heavily affect the performance and power consumption of the system. A variety of interconnection schemes is currently in use, including crossbars, rings, buses, and networks-on-chip [1]. Of these, the latter two have been dominant in the research community.

The shared-buses are among the most widely used communication architectures in systems-on-chip. The main advantages of shared-bus architectures include simple topology, low cost, and extensibility. Several companies have developed their own on-chip bus architectures, such as CoreConnect, AMBA, and Silicon MicroNetworks [1], [2]. An overview of SOC bus architectures is available in [3]. The main problem of shared-bus communications is that the performance of a bus decreases significantly when the bus size (i.e., the number of modules on a bus) increases. This is the so-called scalability problem. Because a bus can be used by only one module at the time, the available bandwidth of one module decreases significantly as the bus size increases. The bandwidth can be improved by hierarchical bus architecture [1] where multiple buses are connected with each other through bridges. However, hierarchical bus architectures may suffer from long communication latency for inter-bus communications.

To overcome the above mentioned limitations, the attention has shifted towards the other popular interconnection architecture: network-on-chip (NOC). This communication scheme assumes that the data is packetized and then transmitted within a chip through a network. NOCs keep their scalability, just like traditional macro networks, but suffer from variable delay as a result of their multi-hop communication nature, and rapid performance deterioration at high data rates. Furthermore, incorporating more sophisticated routing algorithms and priority schemes dramatically increases the complexity of the routers in NOC systems. This, in turn, incurs significant area, power, and latency overhead, since each router is replicated as many times as there are processing elements.

Code-division multiple-access (CDMA) has recently been proposed as a new interconnection mechanism for next generation embedded systems. Compared to a conventional TDMA-based bus, a CDMA-based bus has better features concerning the channel’s isolation and continuity in time domain since channels are divided by the spreading codes [4], [5]. CDMA technology relies on the principle of codeword orthogonality, such that when multiple code words are summed, they do not interfere completely with each other at every point in time, and can be separated without loss of information [6].

This paper focuses on the code-division multiple-access scheme and introduces a CDMA wrapper-based shared bus for SOC. Therefore, we describe (in VHDL), implement (in FPGA and ASIC), and verify (by VHDL testbench simulations) a pair of CDMA bus wrappers (master and slave) that can be used in SOC designs. The benefits and drawbacks of the different bus architectures, designs, and respective implementations are also discussed.

II. WRAPPER-BASED VS STANDARD ON-CHIP COMMUNICATION

The interfacing of IP blocks that use different communication protocols presents one of the major problems in SOC design. Non-standard and custom interfaces make integration a time-consuming task. This problem arises due to IP reuse in design flow process, where an IP core that interfaces with a particular on-chip communication protocol (e.g. IBM CoreConnect [7]) is reused in another design that uses another on-chip communication protocol (e.g. ARM AMBA [8]). Since the interface signals for different protocols are usually different in number, as well as in terms of functionality and timing, IPs prepared for one on-chip communication protocol cannot be easily integrated into a system using another communication protocol. The standard solution of this problem relays on in advance designed and available different core versions for specific on-chip buses and communication protocols. Due to a large number of commercially used on-chip buses, it is very hard or almost impossible to have at disposal (in a core library) the IP core version for each and every on-chip bus and protocol.
The alternative solution for efficient integration of IP cores in SOC is use of the hardware wrappers [9], [10], [11]. A wrapper presents the interface logic between the IP core and its SOC environment. The main wrapper function is conversion of the IP’s interface protocol to a standardized interface protocol.

Therefore, the wrapper-based approach introduces a promising communication technology that separates the communication logic from the IP cores thereby avoiding the connectivity problems related to physical bus protocols. It is based on use of the IP core interface protocol, which is independent of the physical bus protocol, and hardware wrappers that handle the core-to-core communication. Hence, IP cores complying with the interface protocol can be integrated into SOC using the corresponding wrapper. However, attaching even a simple wrapper increases the access latency, so any and every wrapper must include the optimization hardware [9], [10]. At cost of increased hardware overhead, a wrapper can be introduced aiming to achieve an efficient one-to-one protocol conversion between the protocol of IP core and that of on-chip bus considered.

The advantages of the wrapper-based approach in comparison with the standard approach are illustrated in Figure 1. However, we go further and propose a wrapper-based bus that is capable of data transfer from/to IP cores using different protocols and simultaneous multiple master-slave communication.

Figure 1. Two different approaches to communication of IP cores on chip: Standard approach (Solution_1) and wrapper-based approach (Solution_2)

III. MULTIPROCESSOR SYSTEM-ON-CHIP BASED ON CDMA SHARED BUS

In order to develop a solution for wide range of embedded applications, which requires IP core reusability, efficient core interfacing, multiple master-slave connections, and moderate communication performance, we propose the CDMA-coded wrapper-based interconnection as a shared SOC bus.

The wrapper is used as an interface between the CDMA shared bus and IP core connected to it. Two types of the wrappers can be identified: master and slave. A master wrapper is located between the arbiter and CDMA physical interconnect, while a slave connects the CDMA bus with a memory or peripheral module. In our proposal, only bus lines that carry address and data signals are CDMA coded. The architecture of a multiprocessor system based on the CDMA shared bus is shown in Figure 2:

a) The central communication component is the CDMA shared system bus (CSSB). It is composed of three buses: the CDMA coded data bus (DATA_CDMA), the CDMA coded address bus (ADR_CDMA), and the uncoded control bus (Control_CDMA).

b) The master bus wrapper (BW_CPU) converts the internal core data and address signals into CDMA coded bus signals. In all standard solutions (AMBA, CoreConnect, STBus, etc.) the data transfer protocol over a system bus is mainly defined by the timing of control signals and not by the timing of address and data signals. This fact allows us to integrate, with minimal modifications, our approach into any already developed bus protocol. Namely, only signal lines that are used for address and data transfer are CDMA coded, while signal lines that belong to the Control_bus remain unchanged. The penalty that we pay relates to an increased latency, while the benefit is a decreased number of data and address bus lines.

c) The bus arbiter (BA) shown in Figure 2 consists of two functional blocks: the arbiter select logic (ASL) and the arbiter control logic (ACL). According to the bus priority assignment algorithm, the ACL’s output defines which CPU bus (consisting of the DATA_CPU, ADR_CPU, and Control_CPU) bus will drive BW_CPU.

d) Each memory (MEM_i, i=1 ... m) or peripheral (PER_j, j=1 ... k) block is connected to CSSB via the corresponding slave bus wrapper (BW_MEM_i or BW_PER_j). During the write operation, this wrapper converts the CDMA coded bus signals into the memory/peripheral internal signals. Contrary, during the read operation, it converts the internal signals of the memory/peripheral into the CDMA coded bus signals.

e) The wrappers (BW_MEM_i and BW_PER_j) together with the memory/peripheral blocks (MEM_i and PER_j) form the hardware structure of the corresponding memory/peripheral cores (IPMi and IPj).

f) The central processing unit (CPU_w, w=1 ... p), the bus arbiter (BA), and the master bus wrapper (BW_CPU) form the hardware structure of the CPU core. In order to minimize the system hardware, the BA and BW_CPU components are shared for all CPU cores.
IV. CDMA CODING

The main intent of our design is an efficient data and address transmission using the CDMA technique. This coding technique allows us to upgrade the wrapper architecture but retaining both its function and protocol conversion logic almost unchanged. In this section, we explain the principle of CDMA coding and the structures of wrapper’s building blocks that perform this activity.

As is pictured in Figure 2, the master and slave IP cores are connected to the CDMA coded bus through two types of wrappers: BW_CPU and BW_MEM/BW_PER. In general, the hardware structures and operation principles of the master and slave bus wrappers are similar. The following six modules (described in more detail in [12]) make the wrapper structure (Figure 3): the bus wrapper control unit (BWCU), the control protocol transfer unit (CPTU), the CDMA data encoder and decoder (DED), the CDMA address encoder (AE), the configuration register (CR), and the clock generator (CG). The peculiarity of the wrapper structure sketched in Figure 3 relates to the implementation of CDMA encoder and decoder blocks (DED and AE). Namely, instead of a classical non-coded data and address bus transfer, DED and AE provide a CDMA binary coded data and address bus transfer.

Four different connections with parallel signal lines grouped in bundles are possible for implementation of the CDMA coding technique (see Figure 4). The outputs of data/address write register (DWR/AWR), denoted as $D_{IR}$/$A_{IR}$/$A_{IR}$, act as...
independent senders and drive in parallel the spreading block serializer ($SBSD_i$/$SBS_i$). The $DWR$, $AWR$, $SBSD_j$, and $SBS_i$ blocks are constituents of the CDMA encoder. Each data and address bit within a bundle is spread into $v$ bits applying a unique $v$-bit orthogonal spreading code (SC) based on the Walsh functions and transferred serially. Each bit of $v$-bit encoded data generated by the spreading block serializer ($SBSD_i$/$SBS_i$) is a data chip. The data chips which come from different senders within a bundle are summed together arithmetically by the adder of the spreading combiner using two's complement number representation, according to their positions in the $v$-bit sequences. Namely, all the first data chips from different senders within a bundle are added together, all the second data chips are added together, and so on. Therefore, after the addition operations, we get a $w$-bit sum value that equals to $\log_2 v + 2$.

Figure 4. Four different connections in respect to the bundle size: a) 4-bit bundles; b) 8-bit bundles; c) 16-bit bundles; and d) single 32-bit bundle

V. FPGA AND ASIC IMPLEMENTATION

In order to evaluate the performance of proposed approach, we have implemented a CDMA-coded wrapper-based shared bus in two different technologies: FPGA and ASIC. The following assumptions have been made:

a) CPU address and data buses are of 32-bit width.  
b) Data transfer over the CDMA coded bus is achieved by using parallel lines grouped into bundles of 4, 8, 16, or 32 bits.  
c) Orthogonal Walsh functions are used for the CDMA encoding.  
d) The wrapper logic is described at RTL level using VHDL.

A. CDMA Wrapper Performance in FPGA Technology

The design has been synthesized, routed, and mapped into FPGA using the Xilinx ISE 9.1 CAD tool. The verification has been performed using testbenches designed for parallel excitation of all bundle links. The results of design implementation in the Spartan2E, Spartan3, Virtex4, and Virtex5 FPGA devices are presented in Table I. It includes the number of equivalent gates which is proportional to the occupied silicon area, the signal propagation time which corresponds to the total latency of a communication channel, and the dissipated power in mW for a given operating frequency. For a given target device (a row of Table I), the corresponding column specifies:

a) Solution, i.e. the type of bundle (see Figure 4);  
b) Number of the spreading code bits needed for encoding the corresponding bundle of parallel lines;  
c) Number of transmission lines of the CDMA coded bus which equivalently corresponds to the number of bus lines for a standard binary-coded 32-bit bus;  
d) Equivalent gate count needed for implementation of the master ($BW_{CPU}$) and slave ($BW_{MEM/BW_{PER}}$) wrappers, i.e. the logic of a wrapper pair;  
e) Total latency which includes the signal propagation through a master wrapper, a slave wrapper and a CDMA physical bus;  
f) Power consumption of a wrapper logic pair (transmitter and receiver) at 10 MHz clock frequency;  
g) Absolute bandwidth achieved at the maximal operating frequency; and  
h) Energy consumed per single byte transfer.

Based on the results presented in Table I, the following conclusions can be drawn:

1. For all design solutions, increasing the width of a spreading code, what is equivalent to decreasing the number of bundles, the number of lines of a CDMA coded bus decreases, but the latency increases.

2. For a master-slave wrapper pair, the equivalent logic gate count has its minimum when the orthogonal spreading code of 16 bits is used.

3. The consumed energy per byte transfer decreases as the number of lines for data transfer decreases, i.e. the $S4$ solution (a 32-bit bundle) in most cases gives the minimal energy consumption. This is direct consequence of the hardware overhead needed for CDMA coding. The energy consumed per byte transfer is relatively low. It is in the range from 0.088 pJ/B (Virtex5 FPGA device and S4 solution) up to 0.608 pJ/B (Virtex4 FPGA device and S1 solution).

4. The communication bandwidth is always highest for the $S1$ solution (a 4-bit bundle) and smallest for the $S4$ solution (a 32-bit bundle), mainly due to larger number of lines (32 versus 7) and shorter chip sequences (4 versus 32). Data transfer rates are relatively high and are in the range from 587.9 MB/s (Virtex5 FPGA device and S1 solution) down to 30.6 MB/s (Spartan3 FPGA device and S4 solution).
Concerning the choice of FPGA device generation, the implementations in Virtex4 and Virtex5 have the superior performance in respect to the latency. For the worst case, the latency ratio between a device of Spartan3 FPGA and a device of Virtex5 FPGA is equal to 130.688/44.448 = 2.94. Therefore, when the signal propagation is critical, a better choice is a newer generation of FPGA devices. From the other hand, the ratio of equivalent logic gate count for the same pair of implementations is equal to 1183/928 = 1.27. Therefore, in respect to the occupied silicon area, the impact of using a new FPGA device generation is not so pronounced as in case of the design latency.

B. CDMA Wrapper Performance in ASIC Technology

For ASIC implementation, the authors have selected the IHP’s 0.25μm CMOS technology [13] and the S4 solution (32-bit bundle) as a low power solution for wireless SOCs.

The standard IHP’s digital design and implementation flow [14] starts with verification of the VHDL description of a design at RTL level. Then the Synopsys tools [15] are used to produce a gate-level equivalent design with specified timing constraints, i.e. to synthesize it into a target standard cell library. At the end of this synthesizing step, a Verilog net-list and a standard delay format (SDF) file are generated.

After functional and timing verification of the generated Verilog gate-level net-list, a floorplan is created using the Cadence First Encounter layout tool [16]. The design layout has been produced using a standard sequence of the back-end process steps: power planning, placement, clock tree generation, routing and verification of geometry. The chip layout is shown in Figure 5.

Main features of the CDMA wrapper ASIC implementation are summarized in Table II. The equivalent gate area is 21 μm² and much smaller than the equivalent gate area of FPGA devices used. Therefore, a direct comparison of the equivalent gate count of the FPGA and ASIC implementations is not possible. The data show that the silicon area, as well as the estimated power consumption of the ASIC implementation is small.

<table>
<thead>
<tr>
<th>Target device</th>
<th>Solution</th>
<th>Number of SC bits</th>
<th>Number of lines</th>
<th>Equivalent gate count</th>
<th>Total latency (ns)</th>
<th>Power consumption (mW/10MHz)</th>
<th>Absolute bandwidth (MB/s)</th>
<th>Energy per byte transfer (pJ/B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan2E</td>
<td>S1</td>
<td>4</td>
<td>32</td>
<td>1912</td>
<td>15.436</td>
<td>22.40</td>
<td>259.1</td>
<td>0.560</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>8</td>
<td>20</td>
<td>1008</td>
<td>29.136</td>
<td>11.12</td>
<td>137.3</td>
<td>0.278</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>16</td>
<td>12</td>
<td>884</td>
<td>58.176</td>
<td>7.52</td>
<td>68.7</td>
<td>0.188</td>
</tr>
<tr>
<td></td>
<td>S4</td>
<td>32</td>
<td>7</td>
<td>1186</td>
<td>120.992</td>
<td>12.76</td>
<td>33.1</td>
<td>0.319</td>
</tr>
<tr>
<td>Spartan3</td>
<td>S1</td>
<td>4</td>
<td>32</td>
<td>1960</td>
<td>17.16</td>
<td>5.52</td>
<td>227.2</td>
<td>0.138</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>8</td>
<td>20</td>
<td>972</td>
<td>36.336</td>
<td>2.68</td>
<td>110.1</td>
<td>0.067</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>16</td>
<td>12</td>
<td>896</td>
<td>65.392</td>
<td>1.60</td>
<td>61.2</td>
<td>0.040</td>
</tr>
<tr>
<td></td>
<td>S4</td>
<td>32</td>
<td>7</td>
<td>1183</td>
<td>130.688</td>
<td>1.32</td>
<td>30.6</td>
<td>0.033</td>
</tr>
<tr>
<td>Virtex4</td>
<td>S1</td>
<td>4</td>
<td>32</td>
<td>2176</td>
<td>7.752</td>
<td>24.32</td>
<td>516.0</td>
<td>0.608</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>8</td>
<td>20</td>
<td>996</td>
<td>14.36</td>
<td>12.68</td>
<td>278.5</td>
<td>0.317</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>16</td>
<td>12</td>
<td>908</td>
<td>27.184</td>
<td>6.72</td>
<td>147.1</td>
<td>0.168</td>
</tr>
<tr>
<td></td>
<td>S4</td>
<td>32</td>
<td>7</td>
<td>1189</td>
<td>54.336</td>
<td>4.17</td>
<td>73.6</td>
<td>0.104</td>
</tr>
<tr>
<td>Virtex5</td>
<td>S1</td>
<td>4</td>
<td>32</td>
<td>2064</td>
<td>6.804</td>
<td>14.80</td>
<td>587.9</td>
<td>0.370</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>8</td>
<td>20</td>
<td>944</td>
<td>11.52</td>
<td>11.28</td>
<td>347.2</td>
<td>0.282</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>16</td>
<td>12</td>
<td>928</td>
<td>22.256</td>
<td>5.86</td>
<td>179.7</td>
<td>0.146</td>
</tr>
<tr>
<td></td>
<td>S4</td>
<td>32</td>
<td>7</td>
<td>928</td>
<td>44.448</td>
<td>3.54</td>
<td>90.0</td>
<td>0.088</td>
</tr>
</tbody>
</table>

Figure 5. CDMA wrapper in the IHP’s ASIC technology
VI. CONCLUSION

The classical approach for connecting modules on a single chip uses a shared bus. Systems based on a shared bus are easy to implement, test, and understand. Unfortunately, they have some significant drawbacks. First of all, their bandwidth is very limited. Due to high capacity load, their maximum clock frequency is very limited as well. Finally, they apply the TDMA transfer protocol which is not capable of multiple data transfer at the same time.

In this paper, the authors have described an efficient technique for simultaneous data transfers over a CDMA-coded wrapper-based bus. This technique has been implemented for 4-, 8-, 16-, and 32-bit address and data buses in FPGA and ASIC technologies. The main benefit of using the CDMA technique relates to decrease of the number of wires on system bus, in average for 50%, while the main disadvantage is increase of the latency of read and write processor cycles. The increased data transfer latency due to CDMA coding is compensated by simultaneous master-slave data transfers. The power consumption is low and decreases as the number of lines for data transfer decreases.

We believe that there is further room for improvement of the proposed on-chip communication technique, especially if:
   a) Multilevel signaling in a form of pulse amplitude modulation (PAM) and pulse width modulation (PWM) is used [17], [18]. This approach reduces the number of signal paths and/or increases the data rate at a cost of increasing power consumption.
   b) Symmetrical CDMA codes with balanced number of zeros and ones are used. It allows to decrease the minimum number of signal paths for one [19].
   c) Parallel processing in CDMA signal generation is implemented. In this way, a higher speed of communication can be achieved at a cost of increasing the number of data transfer lines.

A combination of the PAM, PWM, symmetrical CDMA codes, and parallel processing in CDMA signal generation represents a promising solution.

REFERENCES