A Novel FPGA Architecture and an Integrated Framework of CAD Tools for Implementing Applications

SUMMARY  A complete system for the implementation of digital logic in a Field-Programmable Gate Array (FPGA) platform is introduced. The novel power-efficient FPGA architecture was designed and simulated in STM 0.18µm CMOS technology. The detailed design and circuit characteristics of the Configurable Logic Block, the interconnection network, the switch box and the connection box were determined and evaluated in terms of energy, delay and area. A number of circuit-level low-power techniques were employed because power consumption was the primary concern. Additionally, a complete tool framework for the implementation of digital logic circuits in FPGA platforms is introduced. Having as input VHDL description of an application, the framework derives the reconfiguration bitstream of FPGA. The framework consists of: i) non-modified academic tools, ii) modified academic tools and iii) new tools. Furthermore, the framework can support a variety of FPGA architectures. Qualitative and quantitative comparisons with existing academic and commercial architectures and tools are provided, yielding promising results.

key words: FPGA, circuit design, CAD tools, RTL design, configuration bitstream

1. Introduction

FPGAs have recently benefited from technology process advances to become a significant alternative to Application Specific Integrated Circuits (ASICs). An important feature that has made FPGAs, particularly attractive is that the logic mapping and implementation flow is similar to the ASIC design flow (from VHDL or Verilog down to the configuration bitstream) provided by the industrial sector [1], [2]. However, in order to implement real-life applications on an FPGA platform, embedded or discrete, increasingly performance and power-efficient FPGA architectures are required. Furthermore, efficient architectures cannot be used effectively without a complete set of tools for implementing logic while utilizing the advantages and features of the target device.

Consequently, research has lately focused on the development of FPGA architectures [3]–[6], [8], [9], [33]. Also, many solid efforts for the development of a complete tool design flow from the academic sector have also taken place [6], [9], [10]. The above design groups have focused on the development of tools that can target a variety of FPGA architectures, while keeping the tools open-source.

Despite the above efforts, there is a gap in the complete design flow (from VHDL to configuration bit-stream) provided by existing academic tools. This is mainly due to the lack of an open-source synthesizer and a FPGA configuration bit-stream generation tool. Therefore, there is no existing complete academic system capable of implementing logic specified in a hardware description language in a FPGA, just an assortment of various fine-grain architectures and tools that cannot be easily integrated into a complete system.

In this paper, such a complete system is introduced. The hardware design of an efficient FPGA architecture is presented. Exhaustive circuit-level exploration in terms of power, delay and area at both Configurable Logic Block (CLB) design and interconnection architecture has been applied in order to make appropriate architecture decisions. Particularly, Basic Logic Element (BLE) using gated clock approach is investigated, at CLB level, while at interconnection network level, new research results about the type and sizing of routing switches are presented in 0.18µm STM process. This investigation is mostly focused on minimizing power dissipation, since it is our primary target in this FPGA implementation, without significantly degrading delay and area. Based on these results and for validation purposes, a full-custom 8 × 8 FPGA was realized in 0.18µm CMOS STM technology.

Additionally, a complete toolset is introduced for mapping logic on the FPGA mentioned above is presented, starting from a VHDL circuit description down to the FPGA configuration bitstream. To best of our knowledge, the developed framework is the only one complete design flow in academia and supports a variety of FPGA architectures. Furthermore, it consists: i) non-modified academic tools, ii) modified academic tools and iii) new tools. The FPGA architecture and tools were developed as part of the AMDRELP project [11] and the tools can be run on-line at the AMDREL website [11].

The rest of the paper is organized as follows: Section 2 describes the FPGA hardware platform in detail, while Sect. 3 is a brief presentation of the tools. Section 4 provides a number of quantitative and qualitative comparisons with existing academic and commercial approaches to evaluate
the entire system of tools and platform. Conclusions are further discussed in Sect. 5.

2. FPGA Architecture

The architecture that was designed is an island-style FPGA [5] (Fig. 1). The main design consideration during the realization of the FPGA platform was the power minimization under the delay constraints, while maintaining a reasonable silicon area. The purpose of this paper is to present the entire system of hardware architecture and software tools not to focus on each design parameter in detail. Therefore, the FPGA design parameters, which were selected through exploration in terms of power, delay and area in [12], [13], are briefly described here.

2.1 Configurable Logic Block (CLB) Architecture

CLB architecture design is crucial to the CLB granularity, performance, and power consumption. The proposed CLB consists of a collection of Basic Logic Elements (BLEs), which are interconnected by a local network (Fig. 2). A number of parameters have to be determined: a) the number of the Look-Up Table (LUT) inputs, $K$, b) the number of BLEs per CLB (cluster size), $N$ and c) the number of CLB inputs, $I$.

LUT Inputs ($K$). The LUT is used for the implementation of logic functions. It has been demonstrated in [32] that a 4-input LUT lead to the lowest power consumption for the FPGA, providing an efficient area-delay product.

Cluster Size ($N$). The Cluster Size corresponds to the number of BLEs within a CLB. Taking into account mostly the minimization of power consumption, our design exploration proved that a cluster size of 5 BLEs leads to the minimization of power consumption (Fig. 2) [12].

CLB Inputs ($I$). An exploration for finding the optimal number of CLB inputs, which provides 98% utilization of all the BLEs [8], results in an almost linear dependency with the number of LUT inputs, and the cluster size, considering the formula:

$$I = \frac{K}{2} \times (N + 1)$$

2.2 Circuit Design

The CLB [12], [13] was designed at transistor level in order to obtain the maximum power savings. It is well known that the minimization of the effective circuit capacitance leads to low power consumption. This is achieved by using minimum-sized transistors, at the cost of delay time. Power consumption minimization involves some techniques such as logic threshold adjustment in critical buffers and gated clock technique. Simulations were performed in Cadence framework [14] using 0.18 µm STM technology.

Table 1 shows the gains achieved by the clock gating technique at CLB level. As shown, the gated clock signal achieves a 83% energy consumption reduction when all the flip-flops (FFs) are “OFF” and a quite smaller increase in energy when one or more FFs are “ON”. The conclusion that the adoption of the gated clock at the CLB level is reasonable when the probability of all FFs in the CLB to be “OFF” is higher than 1/3 is derived from these results.
LUT and Multiplexer Design. The 4-input LUT is implemented by using a multiplexer (MUX), as shown in Fig. 3. The main difference from a typical MUX is that the control signals are the inputs to the LUT and the inputs to the multiplexer are stored in memory cells (S0-S15). LUT and MUX structures with the minimum-sized transistors were adopted, since they lead to the lowest power consumption without degradation in delay. Transistors of minimum size are also used for the 2-to-1 MUX at the output of the BLE.

D-Flip/Flop. A significant reduction in power consumption can be achieved by using Double Edge-Triggered Flip-Flop (DETFF), since it maintains the data throughput rate while working at half frequency. Thus, the power dissipation is halved. Five alternative implementations of the most popular DETFFs in literature were designed and simulated in STM 0.18\(\mu\)m process, in order to determine the optimal one. The one that was finally used is a modified version of the FF proposed in [15], using nMOS transistors instead of transmission gates, because it exhibits low power consumption.

2.3 Interconnect Network Architecture

A RAM-based, island-style interconnection architecture [5], [33] was designed; this style of FPGA interconnect is also employed by Xilinx [1], Lucent Technologies [16] and the Vantis VF1 [17]. More specifically, the logic blocks are surrounded by vertical and horizontal metal routing tracks, which connect the logic blocks, via programmable routing switches. These switches contribute significant capacitance and combined with the metal wire capacitance are responsible for the greatest amount of dissipated power. Routing switches are either pass transistors or pairs of tri-state buffers (one in each direction) and allow wire segments to be joined in order to form longer connections [18]. The effect of the routing switches on power, performance and area was explored in [6].

Alternative configurations for different segment lengths and for three types of the Switch Box (SB) [6], namely Disjoint, Wilton and Universal were explored. A number of ITC benchmark circuits [19] were mapped on these architectures and the energy, delay and area requirements were measured. Another important parameter is the routing segment length. A number of general benchmarks were mapped on FPGA arrays of various sizes and segment lengths and the results were evaluated [12], [13].

Figure 4 shows the energy \times delay products (EDPs) for the three types of SB and various segment lengths. For small segment lengths Disjoint and Universal SBs exhibit almost similar EDPs with the Disjoint topology being slightly better. Also, the lower EDP results correspond to the L1 segment length, meaning that the track has a span of one CLB.

Exploration results for energy consumption, performance and area for the Disjoint switch box topology for various FPGA array sizes and wire segments, are shown in Figs. 5-7, respectively. Based on the above exploration results, an interconnect architecture with the following features was selected:

- Disjoint Switch-Box Topology with \(F_s=3\) [12].
- Segment Length \(L1\) [13].
- Connection-Box (CB): Connectivity equal to one (\(F_c=1\)) for input and output Connection Boxes [12], [13].
- Full Population for Switch and Connection Boxes.
- The size of the CB outputs and SBs transistors is \(W_n/L_n = 10 \times (0.28/0.18)\) [13].

The clock network features H-tree topology and low-swing signaling [13]. The circuits of low-swing signaling driver and receiver are shown in Fig. 8.
2.4 Circuit-Level Low-Power Techniques

Since low-power consumption of the FPGA architecture was the dominant design consideration of AMDREL project, a number of circuit-level low power techniques were employed, including the following:

- Double Edge Triggered Flip-Flops.
- Gated clock at BLE level (up to 77% savings)
- Gated clock at CLB level (up to 83% savings)
- Adjustment of the logic threshold of the buffers
- Minimum transistor size for the multiplexers
- Appropriate transistor sizing for buffers
- Selection of the optimal FF structure for performance and power consumption
- Configuration compression using decoders at CLB and FPGA level
- Low-swing signaling (up to 33% savings on the interconnect network, 47% on the clock signal)
- Minimum width-double spacing in the metal routing tracks
- Interconnection network is realized using the lowest capacitance 3rd metal layer.

Detailed information can be found in [11]–[13].

2.5 Configuration Architecture

The proposed configuration architecture consists of the following components: i) the memory cell, where the programming bits are stored, ii) the local storage element for each tile (a tile consists of a CLB with its input and output connection boxes, iii) a Switch Box plus the memory for its configuration) and iv) the decoder which controls the configuration procedure of the whole FPGA.

Memory cell

The memory cell which is used in the configuration architecture is based on a typical 6T memory cell with all transistors having minimum size. The written data are stored in cross-coupled inverters. Transition gates were used instead of pass transistors because of their stability. The memory cell is provided with a reset mechanism to disable the switch to which it is connected. This prevents the short-circuit currents that can occur in an FPGA, if it is operated with unknown configuration states at start-up. The memory cell can only be written into; the contents cannot be read back. That is why it is sufficient to have a simple latch to store the configuration.

Configuration Element Architecture

Each tile includes a storage element in which the configuration information of the tile is stored. Assuming an 8 × 8 FPGA physical implementation, the configuration element has 480 memory cells because the tile requires 465 configuration bits. The array of the memory cells is 30 columns and 16 rows. The 16 memory bits of a row compose a “word”. During the write procedure the configuration bits are written per “word” because we have a 16-bit write configurations bus. A 5-to-30 decoder is used in order to control which
“word” will be written each time. The 5-inputs of the decoder are connected to the address bus. The structure of the configuration element is shown in Fig. 9.

The decoder was implemented by using 5-input NAND gates and 2-inputs NOR gates because of the small number of inputs. There is also a chip select signal. The NOR gates are used in order to idle the decoder when the chip select has value “0”. A pre-decoding technique was not used because of the increased area and power consumption that it produces.

The configuration architecture of an $8 \times 8$ FPGA array specifications are summarized as follows:

- 4.2 Kb size
- 16-bits data bus
- 12-bits address bus
- 1.4 ns delay for writing a row of 16 memory cells
- 2100 cycles for entire FPGA configuration
- Independent configuration of each tile, allowing partial and dynamic reconfiguration

The layout of a single tile can be seen in Fig. 10.

2.6 FPGA Physical Implementation

A prototype full-custom FPGA was designed in a 0.18 $\mu$m STM process technology. The prototype features:

- $8 \times 8$ array size (320 LUTs, 320 FFs, 96 I/Os)
- 1.8 volts supply voltage
- $4.86 \times 5.28$ mm$^2$ area
- 6 metal layer assignment
  - metal1: Short Connections, Power supply
  - metal2: Short, Intra-cluster, Inter-cluster connections, buses, ground supply
  - metal3: Intra-cluster, Main interconnections
  - metal4: Clock signal, Configuration
  - metal5: Configuration
  - metal6: Configuration
- 2.94 $\mu$sec configuration time

3. Proposed Design Framework

Equally important to an FPGA platform is a tool set, which supports the implementation of digital logic on the proposed FPGA. Therefore, such a design flow was realized. It comprises a sequenced set of steps employed in programming an FPGA chip, as shown in Fig. 11. The input is an RTL-VHDL circuit description, while the output of design flow is the bitstream file that can be used to configure the FPGA. Three different types of tools comprise the flow: i) non-modified existing tools, ii) modified existing tools, iii) and new tools. It is the first complete academic design flow beginning from an RTL description of the application and producing the actual configuration bitstream. Additionally, the
The proposed tool framework can be used in architecture-level exploration, i.e. in finding the appropriate FPGA array size (number of CLBs) and routing track parameters (SB, CB, etc.) for the optimal implementation of a target application. The tools are available at the AMDREL website [11].

All tools can be executed both from the command line and Graphical User Interface (GUI). It should be noted, that the proposed design framework possesses the following attractive features:

- Source description in C/C++ language
- Linux Operating System
- Input format: RTL VHDL, Structural VHDL, EDIF, BLIF
- Output: FPGA Configuration Bitstream
- Implementation Process Technology Independence
- Portability (e.g. i386, SPARC)
- Minimum requirements: x486, 64 MB RAM, 30 MB HD
- Modularity: each tool can run as a standalone tool
- Graphical User Interface (GUI)
- Capability of running on a local machine or through the Internet/Intranet
- Power Consumption and Area Estimation

The following paragraphs provide a short description of each tool.

**VHDL Parser**

VHDL Parser [20] is a tool that performs syntax checking of VHDL input files.

*Input: VHDL code.*

*Output: Syntax check message.*

*Usage: This tool is used to check the correctness of the VHDL file compared to the VHDL-93 standard [21].*

**DIVINER**

Democritus University of Thrace RTL Synthesizer (DIVINER) is a new software tool that performs the basic functions of the RTL synthesis procedure. It converts a VHDL description to an EDIF format netlist, similar to the one produced by commercial synthesis tools such as Leonardo [22] and Synplicity [23].

At present, DIVINER supports a subset of VHDL as all synthesis tools do. DIVINER supports virtually any combinational and sequential circuit, but the combinational part should be separated in the code from the sequential part. In other words, combinational logic should not be described in clocked processes. This imposes no limitations on the digital circuits that can be implemented; it simply may lead to slightly larger VHDL code. DIVINER does not presently support enumerated types in state machines.

DIVINER only performs a partial syntax check of input VHDL files, and therefore, the input files should be compiled first using any VHDL simulation tool, commercial (Modelsim) or open-source (FreeHDL). Additionally, at this stage, DIVINER does not perform Boolean optimization. This task can be done by the SIS optimization tool [27].

DIVINER outputs a generic EDIF format netlist, which can then be used with technology mapping tools in order to implement the digital system in any ASIC or FPGA technology and not necessarily the proposed FPGA hardware platform. More info about the DIVINER, can be found in the tool manual [24].

*Input: VHDL code.*

*Output: EDIF netlist (commercial tool format).*

*Usage: The DIVINER tool is used as a synthesizer of behavioral VHDL language.*

**DRUID**

Democritus University of Thrace EDIF to EDIF translator (DRUID) is a new tool that converts the EDIF format netlist produced by a commercial synthesis tool or DIVINER to an equivalent EDIF format netlist compatible with the next tool of the design flow.

DRUID [24] serves a threefold purpose: i) it modifies the names of the libraries, cells etc, found in the input EDIF file, ii) it simplifies the structure of the EDIF file in order to make it compatible to our tool framework and iii) and it constructs, in the simplest way possible, the cells and generated modules that are included in the input EDIF file and are not found in the libraries of the following tools.

Without DRUID, the hardware architectures that could be processed by the proposed framework would be the ones specified in structural level by using only basic components (inverter, AND, OR and XOR gates of 8 inputs maximum, a 2-input multiplexer, a latch and a D-type FF without set and reset). Moreover, signal vectors are not supported.

*Input: EDIF netlist (commercial tool format).*

*Output: EDIF netlist (T-VPack format).*

*Usage: The DRUID tool is used to modify the EDIF [25]*
output file that is produced during the synthesis step, so that is can be used by the following tools of the design flow.

**E2FMT**

*Input:* EDIF netlist.
*Output:* BLIF netlist.
*Usage:* translation of the netlist from EDIF to BLIF [26] format.

**SIS**

*Input:* BLIF netlist (generic components).
*Output:* BLIF netlist (LUTs and FFs).
*Usage:* SIS [27] is used for mapping the logic described in generic components (such as gates and arithmetic units) into the elements of the proposed FPGA.

**T-VPack**

*Input:* BLIF netlist (gate and F/Fs).
*Output:* T-VPack netlist (LUTs and F/Fs).
*Usage:* The T-VPack tool [10] is used to group a LUT and an F/F to form BLE or a cluster of BLEs.

**DUTYS**

DUTYS (**Democritus University of Thrace Architecture file generator-synthesizer**) is a new tool that creates the architecture file of the FPGA that is required by VPR [10]. The architecture file contains a description of various parameters of the FPGA architecture, including size (array of CLBs), number of pins and their positions, number of BLEs per CLB, plus interconnection layout details such as relative channel widths, switch box type, etc. It has a GUI that helps the designer select the FPGA architecture features and then automatically creates the architecture file in the required format. Each line in an architecture file consists of a keyword followed by one or more parameters. A comprehensive description for the DUTYS parameters, as well as the execution both from command line and through the GUI are stated to the tools manual [24].

*Input:* FPGA features.
*Output:* FPGA architecture file.
*Usage:* Generates the architecture file description of the target FPGA.

**PowerModel (ACE)**

*Input:* BLIF netlist, Placement and routing file.
*Output:* Power estimation report.
*Usage:* The PowerModel tool [9] estimates the dynamic, static and short-circuit current power consumption of an island-style FPGA. It was modified and extended in order to calculate leakage current power consumption.

**VPR**

*Input:* T-VPack netlist (LUTs and F/Fs), FPGA architecture file.
*Output:* Placement and routing file.
*Usage:* placement and routing of the target circuit into the FPGA. VPR [10] was extended by adding a model that estimates the area of the device in mm\(^2\) assuming STM 0.18\(\mu\)m technology.

**DAGGER**

DAGGER (**Democritus University of Thrace e-FPGA bitstream generator**) is a new FPGA configuration bitstream generator. This tool has been designed and developed from scratch. To our knowledge there is no other available academic implementation of such a tool.

DAGGER [24],[28]–[30] is technology independent. This means that it has no constraint about the device design technology. The DAGGER tool supports both run-time and partial reconfiguration, as long as the target device does also. In any case, reconfiguration must be done as efficiently and as quickly as possible. This is in order to ensure that the reconfiguration overhead does not offset the benefit gained by hardware acceleration. Using partial reconfiguration can greatly reduce the amount of configuration data that must be transferred to the FPGA device.

The DAGGER tool flowchart is shown in Fig. 12. As any other program it takes as input the appropriate files and the user parameters. The main steps at the DAGGER tool execution are the bitstream generation, the device initialization, the FPGA configuration and finally, the check about the successful FPGA programming.

The files which are fed to DAGGER tool are: (i) The output from T-VPACK defines the connection of the CLB pins and whether the FF are used in each BLE, (ii) The output from PowerModel provides the LUT programming for each BLE, (iii) the DUTYS tool output determines the FPGA channel width, the switch box topology, as well as the pins topology around the CLB and (iv) the VPR output determines both the location of each BLE to the FPGA array and the routing for all nets.

DAGGER also features the bitstream reallocation technique. This gives DAGGER the ability to defrag the reconfigurable device. In addition to that, the compression that is applied to the bitstream file minimizes the required memory size for storing the FPGA configuration. Another
feature is the error detection which is important whenever there is a non-zero chance of configuration data being corrupted during download to the device. Cyclic Redundancy Checking (CRC) value calculation is used to detect errors and generate an error condition while cancelling the module execution, preventing in this way any damage to the device. Furthermore, important feature is the read-back technique. This feature allows to the programmer to debug successfully any extension to DAGGER, as it reads all the data from the FPGA device back in the internal configuration memory.

The DAGGER output file can be encrypted for security reasons concerning both the FPGA device architecture, as well as the application running on it. Encryption ensures the protection of configuration data from unauthorised examination and modification.

As it is mentioned, DAGGER could handle both runtime and partial reconfiguration types, if they are supported by the target device. Using the selective reconfiguration can greatly reduce the amount of configuration data that must be transferred to the FPGA device.

The partial reconfiguration steps of the DAGGER algorithms are shown in Fig. 13. The DAGGER tool could use two possible approaches in order to generate the partial reconfiguration bitstream, each one with advantages and disadvantages.

In the first technique, every time a reconfiguration is required, the whole bitstream have to be regenerated. Then the existing and the new bitstream are correlated. The correlation output corresponds to the bitstream from the new component, which has to be uploaded into FPGA. In order to regenerate the whole initial bitstream again, we have to correlate one more time the modified bitstream with the bitstream that corresponds to the module. Regarding with the second approach, the bitstream is generated only for the CLBs that have to be reprogrammed and then it is placed into the FPGA. This step is quite similar to the placement problem. The algorithm keeps a map with all the CLBs (programmed or not). The FPGA resources that are placed perimetrical to the array may be reserved for use by the DAGGER tool algorithm or not. If so, this guarantees that all the bitstreams will fit into the array. The disadvantage is the waste of valuable resources.

**Input:** PowerModel output file, Placement and Routing file, FPGA architecture file, T-VPack netlist.

**Output:** FPGA configuration bit stream file.

**Usage:** The DAGGER tool is used to generate the bitstream file.

**Graphical User Interface**

The Graphical User Interface (GUI) provides to the designer with the opportunities to easily use all (or some of the tools) that are included in the developed design flow. It consists of six independent stages: i) the File Upload, ii) the Synthesis, iii) the Format Translation, iv) the Power Estimation, v) the Placement and Routing and vi) the FPGA configuration stage. Until now, there is no other academic imple-

![Fig. 13 Partial reconfiguration flowchart.](image)

4. Comparisons

A complete FPGA system (H/W and S/W) includes a plethora of interdependent parameters, e.g. number of CLBs, LUT size, SB type, etc. On the one hand, we tried to qualitatively evaluate the tool framework by comparing the features it provides with the corresponding features (or lack thereof) of other commercial and academic tool frameworks. On the other hand, quantitative experimental results on different circuit benchmarks were obtained for FPGAs with similar resources with commercial ones.

4.1 Qualitative Comparisons

Qualitative comparisons in terms of provided features among the proposed, XILINX [1], TORONTO [6] and ALLIANCE [31] tool frameworks are provided in Table 2. The symbol + indicates that the corresponding feature is available in the design framework, while the symbol – indicates that the specific feature is not supported by the design framework. The symbol × indicates that the corresponding feature is not provided, but not necessarily for the completeness of that framework either.

Table 2 shows that the proposed design framework provides implementation from as high-level a description as possible (RTL) down to the FPGA configuration file, while it also provides power consumption estimation, and configuration bitstream generation which the other academic frameworks do not. It also features a GUI (which academic frameworks do not) and remote access to it (which no other framework, commercial or academic) does. The only limi-
Table 2 Qualitative comparison among tool frameworks.

<table>
<thead>
<tr>
<th>Feature</th>
<th>proposed</th>
<th>[1]</th>
<th>[6]</th>
<th>[31]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Format</td>
<td>VHDL</td>
<td>VHDL</td>
<td>BLIF</td>
<td>VHDL</td>
</tr>
<tr>
<td>Synthesis</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Format Synthesis</td>
<td>+</td>
<td>×</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Power Estimation</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Area Estimation</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Architecture Description</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Placement</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Routing</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Bitstream Generation</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Partial Reconfiguration</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Back Annotation</td>
<td>−</td>
<td>+</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>GUI</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Remote Access to GUI</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>User Manual</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Operating System</td>
<td>Linux</td>
<td>Solaris/Windos</td>
<td>Solaris</td>
<td>Linux</td>
</tr>
</tbody>
</table>

tations of the proposed framework are that it does not currently support back-annotation, but no other academic tool frameworks do either.

It is evident that the proposed tool framework is the most complete academic tool framework, and is at least in terms of provided features comparable with commercial tools. It contains the only known academic implementation of a configuration bitstream generation tool. Additionally, the remote access to GUI feature allows the user to run the framework without even having the tools installed in his/her own computer.

4.2 Quantitative Comparisons

Various benchmarks from ITC99 [19] (part of the MCNC benchmarks) were implemented in the proposed FPGA array described previously, using the proposed design framework and in Xilinx devices of similar resources using Xilinx ISE tools. The benchmarks range from a few gates to tens of thousands and include combinational, sequential and Finite State Machines (FSMs) circuits. Benchmarks b01-b11 were mapped to the implemented 8 × 8 FPGA device, while benchmarks b12-b21 were mapped to the smallest fitting array, namely from 18 × 18 to 48 × 48.

Figure 14 shows the number of 4-input LUTs used to implement the same benchmarks in the proposed and Xilinx environments. It can be seen that the resulting number of LUTs in the proposed framework is greater. This is mainly due to the fact that the E2FMT tool libraries do not support many basic modules that had to be added by DRUID described at gate level, which leads to larger netlists and therefore greater number of LUTs. This can only be efficiently remedied if E2FMT is drastically modified.

Figure 15 shows the maximum frequencies obtained by the two frameworks and devices. It can be seen that both frameworks perform similarly, with the proposed one outperforming Xilinx in certain benchmarks, while Xilinx outperforming the proposed one in others. More specifically, up to benchmark b11 which is in the order of tens of thousands of gates (the benchmarks get progressively larger in gate count), the proposed framework outperforms Xilinx. For larger benchmarks (about a hundred thousand gates) Xilinx performs somewhat better. This is rather due to inherent limitations of the tools than lack of efficiency on the part of the FPGA architecture. More specifically, the main reason for the somewhat greater delay of the proposed system is due to the greater number of LUTs required to implement the same benchmark in the proposed flow, discussed above.

Still, the frequencies achieved by the proposed framework and device are of the same order as the ones reached by Xilinx Virtex devices.

Figure 16 provides power consumption figures for some of the benchmarks mentioned above. It can be seen that the power consumption of the proposed architecture is...
somewhat greater than that of the Xilinx architecture for benchmarks after b14. Once again, this is due to the tool limitations that lead to an increased number of LUTs. Still, it can be seen that the relative increase in power consumption per benchmark is smaller than the relative increase in number of LUTs (35% and 25% respectively in the case of benchmark b20) which confirms the efficiency of the employed circuit-level techniques. In order to improve the power efficiency of the proposed system, the LUT-mapping process of E2FMT and DRUID will have to be improved.

Figure 17 shows the power consumption for a number of benchmarks with and without the employed low-swing scheme, estimated using PowerModel[8]. It can be seen that the power saved by employing the proposed low-swing technique is significant.

Table 3 shows the results from applying the DAGGER strategy for partial bitstream reconfiguration to the proposed FPGA array for a number of benchmarks. The second column gives the number of bits produced by DAGGER. Finally, the last column gives the percentage gain of the DAGGER bitstream file size, compared to the uncompressed bitstream required to configure the optimal array.

5. Conclusions

A novel FPGA architecture (CLB, interconnect and configuration architecture) with low-power features was presented together with complete tool framework for implementing logic in this platform. The proposed system of the FPGA (implemented in 0.18µm STM technology) and tool framework showed promising results when compared with commercial products using a number of benchmarks.

Acknowledgments

This work was partially supported by the AMDREL project IST-2001-34379, funded by the European Commission.

References

Konstantinos Siozios received both his Diploma degree and his M.S. in Electrical and Computer Engineering from the Democritus University of Thrace, Greece in 2001 and 2003, respectively. He is currently working towards his Ph.D. in the VLSI Design and Testing Center in the same University. His research interests include CAD algorithms and tool development as well as low-power VLSI design.

Konstantinos Tatas received his degree in Electrical and Computer Engineering from the Democritus University of Thrace, Greece in 1999. He is expected to receive his Ph.D. in the VLSI Design and Testing Center in the same University by March 2005. He has been employed as an RTL designer in INTRACOM SA, Greece between 2000 and 2003. His research interests include low-power VLSI design of DSP and multimedia systems, computer arithmetic, IP core design and design for reuse.

George Koutroumpezis received his degree in Electrical and Computer Engineering from the Democritus University of Thrace, Greece in 2002, and his M.S. in the VLSI Design and Testing Center in the same University in 2004. His research interests include reconfigurable VLSI design, IP core design and design for reuse.

Ilias Pappas received the B.Sc. degree in Physics and M.Sc. degree in electronics engineering both from the Aristotle University of Thessaloniki, Greece, in 2002 and 2005, respectively, where currently pursuing the Ph.D. degree in analogue circuits design. His current research interests include reconfigurable architecture full custom design and design of analogue blocks using polysilicon thin film transistors.

Vasilios Kalenteridis received the B.Sc. degree in Physics and M.Sc. degree in electronics engineering from the Aristotle University of Thessaloniki, Greece, in 2001 and 2004, respectively, where he currently is pursuing the Ph.D. degree in reconfigurable computer engineering. His current research interests include reconfigurable computing, computer architecture and VLSI design.

Haroula Pournara received the B.Sc. degree in Physics and the M.Sc. degree in electronics engineering from the Aristotle University of Thessaloniki, Greece, in 2001 and 2004, respectively.

Nikolaos Vassiliadis received the B.Sc. degree in Physics and the M.Sc. degree in electronics engineering from the Aristotle University of Thessaloniki, Greece, in 2001 and 2004, respectively, where he currently is pursuing the Ph.D. degree in reconfigurable computer engineering. His current research interests include reconfigurable computing, computer architecture and VLSI design.

Ilias Pappas received the B.Sc. degree in Physics and M.Sc. degree in electronics engineering from the Aristotle University of Thessaloniki, Greece, in 2002 and 2005, respectively, where currently pursuing the Ph.D. degree in analogue circuits design. His current research interests include reconfigurable architecture full custom design and design of analogue blocks using polysilicon thin film transistors.
Dimitrios Soudris received his Diploma in Electrical Engineering from the University of Patras, Greece, in 1987. He received the Ph.D. Degree in Electrical Engineering, from the University of Patras in 1992. He is currently working as Ass. Professor in Dept. of Electrical and Computer Engineering, Democritus University of Thrace, Greece. His research interests include low power design, parallel architectures, embedded systems design, and VLSI signal processing. He has published more than 130 papers in international journals and conferences. He was leader and principal investigator in numerous research projects funded from the Greek Government and Industry as well as the European Commission (ESPRIT II-III-IV and 5th IST). He has served as General Chair and Program Chair for the International Workshop on Power and Timing Modelling, Optimisation, and Simulation (PATMOS). He received an award from INTEL and IBM for the project results of LPGD #25256 (ESPRIT IV). He is a member of the IEEE, the VLSI Systems and Applications Technical Committee of IEEE CAS and the ACM.

Antonios Thanailakis was born in Greece on August 5, 1940. He received B.Sc. degrees in physics and electrical engineering from the University of Thessaloniki, Greece, 1964 and 1968, respectively, and the Msc. and Ph.D. Degrees in electrical engineering and electronics from UMIST, Manchester, U.K. in 1968 and 1971, respectively. He has been a Professor of Microelectronics in Dept. of Electrical and Computer Eng., Democritus Univ. of Thrace, Xanthi, Greece, since 1977. He has been active in electronic device and VLSI system design research since 1968. His current research activities include microelectronic devices and VLSI systems design. He has published a great number of scientific and technical papers, as well as five textbooks. He was leader for carrying out research and development projects funded by Greece, EU, or other organizations on various topics of Microelectronics and VLSI Systems Design (e.g. NATO, ESPRIT, ACTS, STRIDE).

Stilianos Siskos was born in 1956. He received the B.Sc. degree in Physics from the Aristotle Univ. of Thessaloniki, Greece, in 1980 and the M.Sc. and Ph.D. degrees in Electronics from the University of Paul Sabatier de Toulouse, France, in 1983. He has been a lecturer at the Polytechnic School of Thessaloniki from 1985 to 1989. He joined the Electronics Laboratory, Physics Dept. of the Aristotle Univ. of Thessaloniki in 1989 as a Lecturer and, he is currently an Associate Professor in the same laboratory. His current research interests include analog integrated circuit design, mixed built-in signal structures, current mode integrated circuit design, sensor interfacing integrated circuits, low energy FPGA design for embedded systems, design of signal processing circuits and low voltage analog integrated circuits. He is a member of the IEEE.

Spiridon Nikolaidis received the B.S. and PhD degrees in electrical engineering from Patras University, Greece, in 1988 and 1994 respectively. Since September 1996 he has been with the Department of Physics of the Aristotle University of Thessaloniki, Greece. He is now an assistant professor in the above Department. His current research interests include high speed and low power design of specific-processor architectures, CMOS gate propagation delay modeling and power consumption modeling. He is author and co-author in about 80 scientific articles in international journal and conference proceedings. He also contributes to a number of research projects funded by European Union and Greek Government.