A complete platform and toolset for system implementation on fine-grain reconfigurable hardware

V. Kalenteridis\textsuperscript{a,\*}, H. Pournara\textsuperscript{a}, K. Siozos\textsuperscript{b}, K. Tatas\textsuperscript{b}, N. Vassiliadis\textsuperscript{a}, I. Pappas\textsuperscript{a}, G. Koutroumpezis\textsuperscript{b}, S. Nikolaidis\textsuperscript{a}, S. Siskos\textsuperscript{a}, D.J. Soudris\textsuperscript{b}, A. Thanailakis\textsuperscript{b}

\textsuperscript{a}Electronics and Computers Division, Department of Physics, Aristotle University of Thessaloniki, P.O. Box 54453, Prespas 14-16, 54006 Thessaloniki, Greece
\textsuperscript{b}VLSI Design and Testing Centre, Department of Electrical and Computer Engineering, Democritus University of Thrace, 67100 Xanthi, Greece

Received 20 October 2003; revised 6 May 2004; accepted 30 September 2004
Available online 11 November 2004

Abstract

In this paper a complete system for the implementation of digital logic in a fine-grain reconfigurable platform is introduced. The system is composed of two parts: the fine-grain reconfigurable hardware platform (FPGA) on which the logic is implemented and the set of CAD tools for mapping logic to the FPGA platform. It is the first such complete academic system. The novel energy efficient FPGA architecture was designed and simulated in STM 0.18 \( \mu \)m CMOS technology. The detailed design and circuit characteristics of the Configurable Logic Block as well as the interconnection network are determined and evaluated for energy, delay and area. Concerning the tool flow, each tool can operate as a standalone program as well as part of a complete design framework, composed by existing and new tools.

\( \textcopyright \) 2004 Elsevier B.V. All rights reserved.

Keywords: Low power FPGA; Interconnect architecture; Energy-efficient CLB architecture; Graphical user interface

1. Introduction and related work

FPGAs have recently benefited from process advances to become significant alternatives to ASICs. An important feature that has made FPGAs particularly attractive is a logic mapping and implementation flow similar to the ASIC design flow (from VHDL or Verilog down to the configuration bitstream) provided by the industrial sector \([1,2]\).

Academia has also shown initiative in the development of fine-grain reconfigurable architectures \([3–9]\). Many solid efforts for the development of a complete tool design flow from the academic sector have also taken place (the University of California at Berkeley \([10]\), the UCLA VLSI CAD Laboratory \([11]\), the FPGA Research Group from Toronto University \([12,13]\), the University of British Columbia in Vancouver \([14]\)). 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 (their source-code is available to the public). The main characteristics of the resulting tools are the requirement of a UNIX Operating System (which is quite expensive), as well as the need for some operating system (OS) knowledge by the designer, as these tools run from the command line. Additionally, the academic sector has focused mainly on FPGA technology mapping as well as placement and routing tools, which are of course essential but are not sufficient to provide a tool flow that starts from an HDL description and creates the actual configuration bitstream as commercial tools do.

2. Motivation and contribution

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 due to the lack of an open-source synthesizer and a FPGA configuration bit-stream generation tool. Also the existing ‘incomplete’ design flows operate in text mode, which means that they have no Graphical User Interface (GUI). Additionally, most of the tools were designed and implemented for different operating systems (SUN OS, Linux, BSD, etc). Finally, there is a lack of appropriate manuals for most of the existing tools that will help the end-user to fully understand the provided design opportunities, as well as the programmer to upgrade or to modify their source code. On the other hand, none of the academic platforms mentioned in the previous section, though efficient, provide the necessary toolset for implementing logic on them, at least from the hardware description language level. 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. An exploration in terms of energy, delay and area at both Configurable Logic Block design and interconnection architecture has been applied in order to make appropriate architecture decisions. Particularly, the appropriate circuit design for the Basic Logic Element (BLE) is investigated, while new research results about the type and sizing of routing switches for the interconnection network are presented. This investigation is mainly focused on minimizing energy dissipation, the primary target of this fine-grain FPGA implementation, without degrading significantly delay and area.

Additionally, a complete tool flow for mapping logic on the FPGA mentioned above is presented starting from a VHDL circuit description down to the FPGA configuration bitstream.

Section 3 describes the proposed architecture and circuit design for CLB. In Section 4, both interconnection architecture and routing switches sizing for different wire lengths, widths and spacings, are determined. Section 5 presents the proposed design flow and the FPGA architecture options the tools support. Finally conclusions and future work are discussed in Section 6.

3. FPGA architecture

In this section the FPGA architecture, which can be programmed using the developed toolset, is presented. The main requirement in the design of this FPGA is energy minimization under the delay constraints keeping a reasonable silicon area.

3.1. Configurable logic block (CLB) architecture

The choice of the CLB architecture is crucial to the CLB granularity, performance, density and power consumption. The CLB structure impacts the interconnect architecture and therefore the characteristics of the FPGA. The available Toronto Academic Tools (T-VPACK, VPR) clearly support cluster-based architecture for the CLB [13]. According to this, CLB consists of a collection of Basic Logic Elements (BLEs) connected as it is shown in Fig. 1. Fig. 1a shows the structure of the BLE, which is formed by a LUT, a D-F/F and a 2-to-1 multiplexer, while in Fig. 1b a cluster of these BLEs form the CLB.

The choice of the CLB architectural parameters (LUT Inputs (K), CLB Inputs (I) and Cluster Size(N)) has been made in order to minimize the energy consumption without degrading delay and silicon area. This means that an optimized value for each of them has been determined, from an energy, delay and area perspective, satisfying our main target. In addition, the proposed CLB design is optimized at both circuit and layout level to further reduce energy consumption. Particularly techniques at circuit level such as appropriate transistor ordering and sizing, threshold voltage adjustment, appropriate tapered buffer sizing etc., are used to minimize energy dissipation. At physical level fingering transistors with large width size, using minimum devices spacing, minimum metal width and double spacing of wires, are resulted in the minimization of the capacitance and thus energy consumption.

LUT Inputs (K). The Look-Up Table (LUT) is used for the implementation of logic functions. It has been demonstrated in [23] that four inputs LUT leads to the lowest energy consumption for the FPGA providing an efficient area-delay product.
CLB Inputs (I). In [17] an appropriate number of CLB inputs which provides 98% utilization of all the BLEs is proposed. It is given by the following expression:

$$I = \frac{(K/2)(N + 1)}{2}$$  \hspace{1cm} (1)

The number of inputs in our CLB follows the above formula. By using less inputs than the possible ones without degrading the hardware utilization results in significant capacitance reduction and thus energy consumption.

Cluster size (N). The Cluster Size corresponds to the number of BLEs in a CLB. Our exploration showed that a cluster size of five BLEs leads to the minimization of energy consumption and the energy-delay product [4]. The common case of four BLEs corresponds to the best area-delay product [13].

3.2. Circuit design

The CLB was designed at transistor level in order to obtain the maximum power savings. It is well known that the low power requirement leads to the minimization of the effective capacitance in the circuits. Minimum transistor sizes (i.e. minimum width) were used for the transistors in the CLB (e.g. MUXEs, LUTs, etc). Simulation results show that the use of minimum width for the transistors leads to a lower capacitance and thus energy consumption but at the cost of a slight increase in the propagation delay. In this work, the 0.18 \( \mu \)m STM technology were used and the simulations were performed in the Cadence framework.

The LUT and multiplexer design. The 4-input Look-Up-Table (LUT) of the BLE was implemented using a multiplexer (MUX), as shown in Fig. 2. The control signals are the inputs to the LUT while the inputs to the multiplexer are driven by configuration memory cells (S0–S15) determining the implemented function. Minimum sized transistors were used for LUTs and MUXes since they lead to the lowest energy consumption without significant degradation in the delay.

The D Flip-Flop. A significant reduction in power consumption can be achieved by using Double Edge-Triggered Flip-Flop (DEIFF), since it keeps the same data rate while working at half frequency. In this way the power dissipation on the clock network is halved. Five alternative implementations of the most popular DEIFFs in the literature were designed and simulated in order to determine the optimum one.

Two versions (defined by the type of the used tri-state inverter, Fig. 3) of the Chung F/F proposed in [20] (Chung1 and Chung2) and that of Llopis F/F [19] (Llopis1 and Llopis2) as well as another DEIFF type proposed by Strollo et al. in [15] were designed and simulated. An appropriate input data stream (Fig. 4) to arise all the possible conditions which creates energy consumption were applied to the F/Fs. The total energy consumed during the application of this input sequence for any F/F type is given in Table 1.
Also, the worst case delay from all the combinations of clock signal and data inputs were determined.

As it is observed the F/Fs which present the most favourable characteristics are the ‘Llopis1’ [19] and the ‘Chung2’ [20]. ‘Chung 2’ F/F has the lowest energy-delay product and ‘Llopis 1’ presents the lowest energy consumption. Appropriate sizing of the transistors was performed to further optimize their characteristics. In Table 2 the simulated results for the two optimized F/Fs, in 0.18 μm STM technology, are listed.

As it is observed, ‘Llopis-1a’ F/F presents the lower energy consumption but ‘Chung-2a’ presents the lower energy-delay product. Although ‘Llopis-1a’ F/F, shown in Fig. 5, does not present the lowest energy-delay product, it has simpler structure leading in smaller area. Therefore it was selected as the optimum solution.

**Logic threshold adjustment in critical buffers.** The use of pass transistor logic causes the problem of voltage drop at the output node of a pass transistor, $V_{dd}-V_{th}$ instead of $V_{dd}$. Consequently, the pmos transistor of the next inverter, which takes as input the logic ‘1’, is not completely “OFF”. This may lead, depending on the threshold voltage of the pmos transistors, to a significant leakage current. To reduce this leakage current the width of the pmos transistor is minimized. In Table 3 the value of leakage current for different transistor sizes is shown.

**Gated clock technique.** Gated clock is applied at BLE and CLB level. It is used to isolate the F/Fs that do not operate from the clock network, reducing the transition activity on the local clock network and thus energy consumption.

(a) **BLE level.** At BLE level when the clock enable, $CLK\_ENABLE$, is ‘0’, the F/F is “OFF” and is not triggered. The circuit structures that are used for simulation are shown in Fig. 6, where the shaded inverters in the chain are set for measuring the effect of the input capacitance of the NAND gate on the energy consumption. For the structure in Fig. 6a the average energy amount consumed for a positive and a negative output transition of the F/F is measured. In case of using gated clock (Fig. 6b) the same measurement is taken, considering both ‘0’ and ‘1’ for $CLK\_ENABLE$ signal. The results are given in Table 4. As it is observed, significant energy savings, about 77%, can be achieved when $CLK\_ENABLE$ is ‘0’ (the D-FF is “OFF”). However, when $CLK\_ENABLE$ is ‘1’ there is a slight increase in energy consumption (6.2%) caused by the larger capacitance of the NAND gate than the inverter’s one.

(b) **Gated clock at CLB level.** A gated clock at CLB level can minimize the energy at the local clock network when all F/Fs of the CLB are idle. In this case, the gated clock inputs of the F/Fs and the local clock network of the CLB are constantly at ‘0’ and no dynamic energy is consumed. The circuit structures that were used to measure the energy savings by applying the gated clock technique, are shown in Fig. 7.

### Table 1
Energy consumption, delay and energy-delay product

<table>
<thead>
<tr>
<th>Cell</th>
<th>Total energy (fJoules)</th>
<th>Delay (ps)</th>
<th>Energy–delay product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chung 1</td>
<td>433.1</td>
<td>163.5</td>
<td>$70.8 \times 10^{-24}$</td>
</tr>
<tr>
<td>Chung 2</td>
<td>457.2</td>
<td>135.3</td>
<td>$61.9 \times 10^{-24}$</td>
</tr>
<tr>
<td>Llopis 1</td>
<td>409.0</td>
<td>217.2</td>
<td>$88.8 \times 10^{-24}$</td>
</tr>
<tr>
<td>Llopis 2</td>
<td>429.7</td>
<td>241.8</td>
<td>$104 \times 10^{-24}$</td>
</tr>
<tr>
<td>Strollo</td>
<td>413.5</td>
<td>270.0</td>
<td>$112 \times 10^{-24}$</td>
</tr>
</tbody>
</table>

### Table 2
Energy consumption, delay and energy-delay product for optimized F/Fs

<table>
<thead>
<tr>
<th>Cell</th>
<th>Total energy (fJoules)</th>
<th>Delay (ps)</th>
<th>Energy–delay product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chung-2a</td>
<td>436.3</td>
<td>138.5</td>
<td>$60.4 \times 10^{-24}$</td>
</tr>
<tr>
<td>Llopis-1a</td>
<td>387.7</td>
<td>194.8</td>
<td>$75.5 \times 10^{-24}$</td>
</tr>
</tbody>
</table>

**Fig. 4.** Input pulses to the Flip/Flops for simulation.
The energy consumption is measured for various conditions. The simulation results are given in Table 5.

As it can be observed the gated clock technique results in a 83% reduction in energy consumption when all the F/Fs are “OFF”, in a 33% increase when only one F/F is “ON”, and in a 29% increase when all the F/Fs are “ON”. From these results it is clear that the adoption of the gated clock at the CLB level is reasonable, as long as the probability of all the F/Fs in the CLB to be “OFF” is higher than 1/3. The final adoption of the gated clock at CLB level is determined by experiments using the physical design of these structures, in order the wire capacitance to be considered.

Selected CLB architecture. Based on the results mentioned in the previous sections and those reported in the literature, a decision for the CLB architecture was made. Consequently, the features of the selected CLB are:

(a) Cluster of 5 BLEs, (b) 4-inputs LUT per BLE, (c) One double edge-triggered Flip-Flop per BLE, (d) Gated Clock at BLE and CLB level, (e) 12 inputs and 5 outputs provided by each CLB, (f) All five outputs can be registered, (g) Fully Connected CLB resulting to the use of 17 to 1 multiplexer for each LUT input, (h) One asynchronous Clear signal for whole CLB and (i) One Clock signal for whole CLB.

Next a comparison of the proposed FPGA architecture with the SPARTAN2 Xilinx FPGA, device xc2s15 (they present similar characteristics), is presented. One combinational and one sequential circuits were implemented on these FPGAs.

<table>
<thead>
<tr>
<th>Proposed FPGA</th>
<th>XILINX SPARTAN2 xc2s15</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Combinational</strong></td>
<td></td>
</tr>
<tr>
<td>Synthesis</td>
<td>Used/avail.</td>
</tr>
<tr>
<td>No. of primary IOs</td>
<td>58/96</td>
</tr>
<tr>
<td>No. of LUTs</td>
<td>148/320</td>
</tr>
<tr>
<td>Clusters</td>
<td>148/320</td>
</tr>
<tr>
<td>CLBs</td>
<td>30/64</td>
</tr>
<tr>
<td>IOs</td>
<td>58/96</td>
</tr>
<tr>
<td>Delay (ns)</td>
<td>19.6</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>7.4</td>
</tr>
<tr>
<td>Signal power (% of total)</td>
<td>44%</td>
</tr>
<tr>
<td>Logic power (% of total)</td>
<td>56%</td>
</tr>
<tr>
<td><strong>Proposed FPGA</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Sequential</strong></td>
<td></td>
</tr>
<tr>
<td>Synthesis</td>
<td>Used/avail.</td>
</tr>
<tr>
<td>No. of primary IOs</td>
<td>23/96</td>
</tr>
<tr>
<td>No. of LUTs</td>
<td>313/320</td>
</tr>
<tr>
<td>No. of latches</td>
<td>74/320</td>
</tr>
<tr>
<td>Clusters</td>
<td>313/320</td>
</tr>
<tr>
<td>No. of clocks</td>
<td>1/1</td>
</tr>
<tr>
<td>CLBs</td>
<td>64/64</td>
</tr>
<tr>
<td>IOs</td>
<td>23/96</td>
</tr>
<tr>
<td>Delay (ns)</td>
<td>25.6</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>5.5</td>
</tr>
<tr>
<td>Signal power (% of total)</td>
<td>32%</td>
</tr>
<tr>
<td>Logic power (% of total)</td>
<td>68%</td>
</tr>
</tbody>
</table>

Fig. 5. Llopis Flip-Flop proposed in [19]
The low power characteristics of the proposed FPGA are obvious. Also, smaller delay times result in higher frequencies.

4. Interconnect network architecture

The island-style interconnection architecture [12] was used for our FPGA since this style is supported by the Toronto academic tools. In this interconnection style, the logic blocks are surrounded by vertical and horizontal metal routing tracks which are connected to compose the interconnect architecture, via programmable routing switches. These switches contribute significant capacitance and with the metal wire capacitance are responsible for the greatest amount of the dissipated energy. Since our primary concern is to minimize energy consumption in this interconnection network, an investigation about the size of the routing switches, when driving wire segments of different length, is needed. However, the delay and the required area should remain into acceptable values. 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 long connections [22].

4.1. Sizing pass transistor routing switches

In this section the best routing pass transistor width is determined. All experimental values have been derived for the STM's 0.18 μm, six metal layers CMOS process.

Fig. 8 shows a typical routing interconnection composed by four wires of logic length of one used for the connection of eight logic blocks (CLBs). The logical length is defined as the number of logic blocks spanned by a routing wire and it is varied over a wide range. It is assumed that the connection box flexibility, $F_c$ (a fraction number determining the percentage of routing tracks in the channel connected to a CLB pin) is equal to 1 for both input and output pins of the CLB (fully connected CLB). This value is considered for the worst-case scenario. The disjoint switch block topology, which gives an $F_s$ of three (which defines the number of routing wires which can be connected via routing switches), is used. This value is used in the most commercial implementations. It is also assumed that the pass transistors, which permit to connect the logic block output pin at each routing track, hang on each track and add extra parasitic capacitance. These transistors have the same size as the other routing switches [13]. The wire is also loaded by the buffers, which drive the data signals into the logic block input pins. In addition it should be noted that the routing wires are laid out in metal 3, because they have the lowest capacitance value among routing metals for the used technology. Moreover, it is investigated if FPGA routing wires benefit from greater than minimum metal width or spacing.

Fig. 9 plots the energy-delay–area product as a function of the pass transistor width for four different wire lengths, in the case where minimum metal wire width and minimum spacing between adjacent metal wires is used.

The plotted curves show that for wire lengths of 1, 2 and 4 logic blocks, transistor size of 10 and 16 times the minimum one are essentially tied for the best energy-delay–area product. However, it is obvious that the best width is 64 times the minimum for wire length of 8, but it leads to an unacceptable silicon area.

The energy-delay–area product as a function of the pass transistor width for different wire lengths is plotted in Fig. 10, for the case which minimum metal wire width and double-spacing is used. Energy-delay–area product is improved in this case, which means that this wire configuration is more efficient than the previous one. This result can be explained considering that an increase in spacing between metal wires decreases the stray capacitance, leading to energy reduction. As it can be shown, the optimum routing pass transistor width is 10 times the minimum one for wire lengths of 1, 2 and 4 logic blocks, while it still remains 64 times the minimum width for wire length of 8. Finally the curves of energy-delay–area product relative to the different widths of the routing switches for double metal wire width and double-spacing, are illustrated in Fig. 11. As it can be shown from the plots, transistor width of 10 times the minimum one yields the best result for energy-delay–area product, for wire lengths of 1, 2, 4 and 16 times the minimum for wire length of 8.
Summarizing, the optimum width from an energy, delay and area perspective is merged to be 10 times the minimum one for wire length of 1, 2 and 4 for all previous cases. The best routing pass transistor size for wire length of 8 is 64 times the minimum one. However, since the switch box area is affected significantly by the routing transistor size, this width would have an obvious increase in its physical area, influencing total FPGA die.

4.2. Sizing tri-state buffer routing switches

In order to determine the best size of tri-state buffers, a procedure identical to that described in the prior section is followed. The configuration of Fig. 8 is used, except that each pass transistor between two routing wire segments is replaced by two tri-state buffers—one in each direction.

In the tri-state buffer sizing exploration, two stages have been used in order to minimize energy-delay–area of the buffer [16]. The width ratio of the first stage is kept constant at 0.68/0.28 ($W_p/W_n$). For the second stage, transistor widths up to 16 times the ones of the first stage are used, in order to explore their effect in energy-delay–area product.

Fig. 12 shows the energy-delay–area product versus the size of the tri-state buffers, for minimum width and minimum spacing of wires. As it can be seen, increasing the buffer size, the product of the performance criteria is improved. The optimum buffer size is found to be twice the minimum one for wire length of 1. Further increase in the size of the buffer causes a delay making it more significant than the decrease in the time it takes to discharge the capacitance of the routing wire. For the rest wire lengths the optimal buffer size closest to the ‘knees’ of the plots is found to be four times the minimum.

Fig. 13 presents the three criteria product in relation to the tri-state buffer size for minimum width double-spacing of wires. In this case an optimal size of two is resulted for logic lengths of 1, 2 and four times the minimum one is obtained for wire lengths of 4, 8, respectively.

Finally Fig. 14 shows again the performance product relative to tri-state buffer size for double width double-spacing of wires. These curves result to the same information about the buffer size as in the previous case, which means that only spacing out the metal wires yields to a better energy-delay–area product. Consequently, it can be stated that the best tri-state buffer size is two or four times the minimum one depending on the exploration criteria. However, it should also be noted that the three performance product values are greater than those of the routing pass transistors found in the previous section. Therefore, pass transistors routing switches with a wire length of 1 and minimum width double-spacing will be used in order to achieve a low energy fine-grain FPGA.

Conclusively, our exploration results [28] for the interconnection network architecture have shown that for an optimum design in terms of energy consumption, an architecture with $F_c = 1$ for input and output CBs, with segment length of one should be used. Energy savings of 25% can be achieved with a small decrease in performance, over the fastest architecture.

Simulation experiments have also shown that the critical path delay, as the signal travels from one register to another one, passing from one interconnection routing track is about 3.3 ns. This corresponds to a maximum operating system frequency of 300 MHz.
5. Proposed design flow

Equally important to the FPGA architecture is a tool flow that supports the implementation of digital logic on the proposed FPGA. Therefore, such a design flow was realized, fulfilling both the needs of experienced designers by providing practical answers to state-of-the-art problems (like logic synthesis and bit stream generation), and novice designers by providing a simple and consistent set of tools.

The proposed design flow comprises a sequenced set of steps employed in programming a FPGA chip. Fig. 15 presents the design methodology. The circuit is first described in VHDL, while the output of the CAD flow is the bit stream file that will be used to program the FPGA. As it is shown in Fig. 15, three different types of tools comprise the flow: (i) non-modified existing tools, (ii) modified existing tools, and (iii) new tools.

Our contribution in the proposed design framework involves the selection of the appropriate tools that form the design flow, the porting of existing tools to the Linux OS, the design and implementation of the new tools (or the modification of existing ones), the development of the Graphical User Interface that could be used through the Internet, and finally the writing and updating of manuals of all the tools that form the design flow.

5.1. Tools of the proposed design flow

In this section, the tools that form the complete design framework are described. All of them can be executed both from the command line and the Graphical User Interface that is presented in detail in the next subsection.

VHDL Parser. This tool is used to check the correctness of the VHDL file compared to the prototype VHDL-93 [24]. When the described circuit is syntactically correct, the output of the program is a confirmative message. On the other side, when the VHDL input is syntactically wrong, the VHDL_Parser [25] apart from returning an error message, it also provides information about the location and the nature of the error. Input: VHDL source. Output: Syntax check message.

DEVINER. The DEVINER tool is used as a synthesizer of behavioral VHDL language. It should be pointed out that there is no other known academic synthesis tool available. The output file is a netlist of gates in EDIF format, which is the same compared to commercial synthesizers. Even though DEVINER can so far only synthesize a circuit description that consists of logic gates and F/Fs, it can be replaced by an existing commercial synthesis tool, such as Leonardo, in order to synthesize circuits with more complex behavioral descriptions. Input: VHDL source, Output: EDIF netlist (commercial tool format).

Druid. The DRUID tool is used to modify the EDIF [27] output file that is produced during the synthesis step, so that it can be used by the following tools of the design flow. So far, there is no other available design tool which can bridge the gap between the EDIF and BLIF format. Without this tool the circuits that will be mapped into the FPGA have to be described directly in BLIF format [26], which renders the advantages that VHDL offers unusable. The modifications made by the DRUID tool concern the structure as well as some keywords of the EDIF file. Input: EDIF netlist (commercial tool format), Output: EDIF netlist (T-VPack format).

E2FMT. This tool is used to translate the netlist from EDIF to BLIF format, as the latter is required by the next tool in the design flow. Based on the bibliography research, the E2FMT tool [18] is more efficient than the other existing tool with the same functionality (edif2blif), as it can handle more complex EDIF files. The translation is based on the IWLS’93 library of the LGSynth’93 benchmark. This technology library has been extended in order to support more complex components. The source code of the tool was modified to embody the PowerModel (ACE) tool within the design flow. Input: EDIF netlist, Output: BLIF netlist.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Simple clock</th>
<th>Gated clock (NAND)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All F/Fs “OFF”</td>
<td>E = 23.1 fJ</td>
<td>E = 3.9 fJ</td>
</tr>
<tr>
<td>One F/F “ON”</td>
<td>E = 24.1 fJ</td>
<td>E = 32.1 fJ</td>
</tr>
<tr>
<td>All F/Fs “ON”</td>
<td>E = 27.8 fJ</td>
<td>E = 35.8 fJ</td>
</tr>
</tbody>
</table>

Table 5
Energy consumption for simple and Gated clock at CLB level

Fig. 8. FPGA routing experiment circuitry.
T-VPack. The T-VPack tool [21] is used to group a LUT and an F/F to form a Basic Logic Element (BLE) or some BLEs to form a Cluster. The advantages of this tool is that it can cooperate with the VPR and DAGGER tools, as well as it is quite easy to modify the source code in order to handle more complex structures as BLEs. In the proposed version T-VPack supports the type of BLE described in Section 3.1. The drawback of this is that any modification to T-VPack must be compatible with the placement and routing tool. Significant part of this work was the determining of the way that this tool could handle user-specified structures of BLEs, in other words the interaction between the T-VPack modifications and the FPGA architecture specification. Input: BLIF netlist (gate and F/Fs) Output: T-VPack netlist (LUTs and F/Fs).

DUTYS. The DUTYS tool generates the architecture file description of the target FPGA. Among the parameters that include this description are the FPGA channels, the logic block, the routing architecture, the timing analysis, and power consumption, which are extracted from simulations. The interaction between this tool and the FPGA architecture design is obvious. The output file will be used both to the VPR and DAGGER tools. To the best of our knowledge, so far no other existing academic tool can do this. Input: FPGA features, Output: FPGA architecture file.

VPR. The VPR tool [21] is used to place and route the target circuit into the FPGA. There are two possible routing...
algorithms: (i) the breadth-first router focuses solely on routing a design successfully, while (ii) the timing-driven router focuses both on achieving a successful route and achieving good circuit speed. The breadth-first router is capable of routing a design using slightly fewer tracks than the timing-driven router (typically 5%). The designs produced by the timing-driven router are much faster, however, $(2x - 10x)$ and it uses less CPU time to route. The interconnect network architecture is a significant factor in the choices that VPR has to make during the routing phase. Combined with PowerModel, it extracts information about the power consumption of the circuit. The source code of this tool has been modified in order the graphic output of the program to be included in the GUI. Input: T-VPack netlist (LUTs and F/Fs), FPGA architecture file, Output: placement and routing file.

PowerModel (ACE). The PowerModel tool [14] estimates the dynamic, short-circuit, and leakage power consumption of an island-style FPGA. The model is flexible enough to target FPGAs with various look-up table (LUT) sizes, interconnect strategies (segment length, switch block type, connection flexibility), cluster sizes (for a hierarchical FPGA), and process technologies. This tool is integrated to the proposed design flow, after porting from Solaris-OS to Linux. In addition to that, the PowerModel tool has been debugged in order to be used in combination with the tools E2FMT and T-VPack. Input: BLIF netlist, Placement and routing file, Output: Power estimation report.

DAGGER. The DAGGER tool is the first academic “open source” bit-stream generation tool. It receives input from PowerModel, DUTYS, T-VPack, and VPR tools. The algorithm that used in order to represent the connection (both the routing channels and inside the BLE) is novel. The FPGA components are parameterized, which allows the tool to target a variety of FPGA architectures. The output bit-stream configuration is in encrypted binary format, as this is more secure both for the FPGA and the circuit implemented on it. As shown in Fig. 16, the FPGA has been encoded as a set of arrays. Part of the tool is the routine that uploads the bit-stream configuration to the FPGA. The DAGGER tool can support partial reconfiguration, if the target FPGA supports it. Finally, it can be used in the same way for programming BLEs or Clusters. Input: Placement and Routing file, FPGA architecture file, T-VPack netlist, Output: FPGA configuration bit stream file.

It should be noted, that the proposed flow possesses the following attractive features:

(i) **Technology Independence.** The proposed CAD flow provides process technology independence in order to allow the designers to easily implement their design in different process technologies (0.18, 0.15, and 0.13 μm).

(ii) **Portability.** The proposed flow has been designed to run on several hardware platforms, from i386 based microcomputers to SPARC stations.

(iii) **Modularity.** Each tool can operate as a standalone program as well as a part of the complete design framework. For this reason, most of the tools support several different standard VLSI circuit description formats (VHDL, EDIF, BLIF).

(iv) **Compactness.** Unlike commercial CAD tools, the proposed CAD framework suits the limited resources of low-cost PCs. For PC operation, the minimum requirements for a Linux system are an i486 PC with 32 Mbytes of memory, appropriate disk storage (350 Mbytes), and graphic capabilities (X-Windows). On the other hand, there is the possibility to use the design flow through the Internet or the CD-ROM. In the first case, the users need to have only a web-browser installed into their PC and a connection to the Internet. On the other hand, if the CD-ROM is chosen, then the user has only to boot the PC with the appropriate Live-CD that includes the LINUX...
Operating System as well as the executables of the tools, and start using the design flow.

(v) *Ease of use.* All the tools as well as the proposed design flow, are simple to use with no experience required at the Linux OS, as the Graphical User Interface (GUI) provides a user-friendly interface. Furthermore, the on-line documentation in combination to the paper and manuals that are provided with each tool, help the non-experienced user to program the FPGA by using the proposed design packages. It should be noted that this tool set in its current version supports the island-style FPGA architecture described in Section 4.

5.2. Graphical user interface

The Graphical User Interface (GUI) provides the designer with the opportunities to easily use all (or some of the tools) that are included in the proposed design flow. The GUI is shown in Fig. 17. It consists of six independent stages, which are the File Upload, the Synthesis, the Format Translation, the Power Estimation, the Placement and Routing, and the FPGA Program stage. Until now, there is no other academic implementation of such a complete graphical design chain. The main GUI advantage is the fact that it is friendly to the non-experienced designer. The end-user does not need to be familiar with the Linux OS, it is possible to run it from a local PC or through the Internet/Intranet, and the source code can be easily modified in order to add more tools. Regardless of the execution (locally or through the network) the proposed interface runs on the web-browser, and can program an FPGA that is attached to the user’s PC. In order to help users with no experience of the GUI, a short description of each tool is displayed every time the mouse pointer is over the corresponding icon.

![Fig. 16. The “encoded” FPGA for the DAGGER tool.](image1)

![Fig. 17. The graphical user interface (GUI) for the tool.](image2)
5.3. Comparisons

Various benchmarks from ITC’99 [28] (part of the MCNC benchmarks) were implemented in the proposed FPGA using the proposed design framework and in Xilinx devices of similar resources using Xilinx ISE 4. The benchmarks range from a few gates to tens of thousands. Fig. 18 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 and Xilinx outperforming the proposed one in others. Still, the frequencies achieved by the proposed framework and device are of the same order as the ones reached by Xilinx Virtex devices.

6. Conclusions

This paper demonstrated the first complete system for implementing digital logic on a fine-grain reconfigurable platform. It includes the design of both the FPGA architecture and the complete design flow (from VHDL to bitstream) consisting entirely of academic tools, which allows the mapping of logic on the presented novel FPGA architecture. The novel FPGA architecture was designed and implemented in STM 0.18 \( \mu \)m CMOS technology. The obtained simulation results prove the attractive features of the proposed architecture. On the other hand, in contrast to the commercial CAD system, the proposed design flow does not require expensive workstations; it can accomplish a FPGA design and is publicly available and very friendly to the non-experienced designers.

Acknowledgements

This work was partially supported by the project IST-34793-AMDREL which is funded by the E.C.

References

on Low Power Electronics and Design, August 1996, Montray, USA.


[27] http://www.edif.org