ハードウェア記述言語によるＡＬＵ設計と試作

川島 由治[1]・渡邊 孝博[2]・棚田 嘉博[2]・守川 和夫[3]

[1]日本電信電話 (株)
[2]知能情報システム工学科
[3]徳山高等工業専門学校

ディジタルシステムの規模と複雑さが増大すると共に、階層設計方式とそれを支援するCADツールが不可欠になっている。中でも、設計工程上流でシステムの動作や構造を「ハードウェア記述言語」で記述し、「回路合成ツール」によって回路を作成する方法が主流となってきた。
そこで我々は、その設計技法を経験しノウハウを蓄積すると共に、限界や問題点を把握することを目的に、ハードウェア記述言語VHDLによるALU設計を試行した。また、設計したシステムを市販の計算機演習用FPGAボードに実装することによって、費用・期間の面で効率良くハードウェア検証が実現できた。

Key Words : *Digital design, CAD, HDL, VHDL, FPGA, KITE board*

1. はじめに

近年、ディジタルシステムは益々大規模化、複雑化しており、ゲートレベルあるいはトランジスタレベルで設計することはもはや不可能になりつつある。代わって登場して来たのが、システムをALU、キャッシュメモリ、浮動小数点演算器などの機能的な構成要素に分割し、そのようなレベルから階層的に設計を進める、いわゆる「トップダウン設計方式」である。このような設計階層は、最上流のアーキテクチャレベルから、機能レベル(動作レベル、あるいは、RTLレベル)、ゲートレベル、トランジスタレベルとつながる。ハードウェア記述言語（HDL：Hardware Description Language）は正にこのような階層設計の上流での設計を支援するために開発されて来たものである。例えば、上位の機能レベルでシステムを記述し、これを高位合成あるいは論理合成と称する回路生成CADに入力すると、下位の論理回路図などが得られるものである。そのような設計方式では、設計者は従来のように回路図を書く必要はなく、システムの機能や動作を高級言語のようなプログラムで記述すればよい。

HDLとしてはここ数年間に複数の提案がなされて来たが、VHDLおよびVerilog-HDLの2種類がIEEE標準として認可され、現在、急速に普及が進んでいる状況である。

本稿では、HDL設計手法が設計回路の規模や種類、設計サイトを問わず不可欠になりつつあることから、その設計技法を経験し、ノウハウを蓄積すると共に、限界や問題点を探っておく必要があると考え、小規模なディジタルシステムの設計を行ったものである。また、設計した結果は単にシミュレーションによって検証するだけでなく、実際のハードウェアで構成して評価することが望ましい。しかし、これをLSI化またはプリント板に実装することは、費用、期間の面で難がある。そこで、我々は計算機教育/演習用として市販されているプロセッサボードを利用し、このボード上に搭載されている書き込み自由のFPGA(Field Programmable Gate Array)に実装することで、設計物のハードウェア試作を行った。

本稿の構成は、第2章でHDLの紹介と、HDLによる設計の流れを示す。第3章では今回の設計対象であるALUを説明する。第4章は設計及び実装の環境と、実際の試作結果を述べる。

2. ハードウェア記述言語による設計

大規模ディジタル・システムの設計は論理合成技術の開発と普及により、ソフトウェアにおける高級言語によるプログラミングと同等な程度にまで、自動化が進められてきた。即ち、高級言語に対応するハードウェア設計記述言語(HDL:Hardware Description Language)により機能設計(演算器やレジスタなどの基本的なハードウェアモジュールとその間のデータの移動、及びその制御条件)を記述しさえすれば、論理合成とレイアウト合成により自動的に回路のマスクパターンを生成することが可能になりつつある。原理的

には、レイアウトに関する知識や論理設計に関する細かい知識を持たなくても、システムの動作を言語で記述できれば、ハードウェアの設計ができる状況になっている。

このハードウェア記述言語は回路の動作や構造を記述するための言語であり、回路設計では従来のように回路図を書くのではなく、この言語によって回路を記述する。

ハードウェア記述言語には、代表的なものとしてVHDL、Verilog-HDL、SFLなどが知られているが、本論文ではVHDLを使用している。

本章では、VHDLの概要とこれによる設計作業を説明する。VHDLの詳細については、既に市販の解説書[1]やマニュアル[2]があるのでそちらを参照されたい。

2.1. VHDL言語

VHDL(VHSIC Hardware Description Language)は1981年に米国防省のVHSIC(Very High Speed Integrated Circuit)プロジェクトによって提案されたVHSIC-HDLが原型であり、デバイス・アーキテクチャに依存しない開発手法として提案された。

1986年に公開され、米国防省関連のASIC設計仕様書に、VHDLの添付が義務付けられた。

その後、IEEE std 1076(1987年)及びIEEE std 1164(1992年)として言語仕様が標準化され、又、VHDLシミュレータやVHDL記述からの回路合成ツールが開発されるに及んで、広く普及しつつあるものである。

(1) VHDLの動作モデルと構造モデル

動作モデルとはモジュールがどのように動作するかを抽象化することである。モジュールの出力をその入力を用いて記述するのであるが、そのモジュールが論理ゲートによってどのように具体的に実現されるかということにはまったく注目しなくてよい。

動作モデルはプログラム言語と類似した言語で記述される。抽象化のレベルにより、レジスタ転送レベル(RTL：Register Transfer Level)の回路動作を記述したり、論理ゲートやフリップフロップの動作を記述したりすることもできる。いずれの場合も、動作モデルという概念によって、具体的な構成方法を限定せずに、モジュールの機能を指定する。

構造モデルとはモジュールを論理ゲートのレベルで設計する手法である。

(2) VHDLによる記述

VHDLは広範な抽象化レベルでデジタルシステムを規定することのできるハードウェア記述言語である。上流の概念設計段階における動作レベルから、設計工程後半の構造記述レベルまでをサポートしている。設計工程中に、各部分の設計が進行する過程で、動作記述と構造記述の混在が許され、階層記述によって複雑な設計にも対応できる。

VHDLではデジタルシステムをモジュール(module)の集まりとして記述する。これらのモジュールは中身に関する記述(ARCHITECTURE)と、他のモジュールとのインターフェイス(ENTITY)を含む。モジュールは論理ユニットを表し、論理構造によって定義する(例えば実際の論理ゲートで記述する)か、あるいはプログラムのように動作を定義する。これらのモジュールはインターフェイスで他のモジュールと接続される。図1に全加算器をVHDL記述した例を示す。

```
LIBRARY WORK,IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

-- Full Adder
ENTITY full_add IS
    PORT (
        x      : IN STD_LOGIC ;
        y      : IN STD_LOGIC ;
        c      : IN STD_LOGIC ;
        sum    : OUT STD_LOGIC ;
        carry  : OUT STD_LOGIC
    );
END full_add ;

ARCHITECTURE behave OF full_add IS
BEGIN

  sum <= (x XOR y) XOR c ;
  carry <= (x AND y) OR (x AND c)
           OR (y AND c) ;

END behave ;
```

図1: 全加算器のVHDL記述例

(3) VHDLの基本的機能

VHDLの基本的な機能で特に重要な点をあげれば以下の通りである。

- デジタルシステムでは、システムを構成する要素の規模が大小を問わず並列に動作する。そのため、VHDLは基本的に全モジュールを並列で動作させる機能を持つ。
- 階層設計をサポートしているので、設計上位レベルを複数の下位レベルに分割して設計できる。こうしたモジュールの階層化によって設計の複雑さを克服することができ、巨大な回路設計も可能である。
- 設計を抽象的な動作あるいは実際の論理構造のいずれによってでも記述できる機能。この動作

記述によって、設計の初期には機能面に専念できる。動作が満足されれば次にいくつかの実装上の構造を選定してゆく工程に移ることができる。

2.2. VHDLによる開発作業の流れ

動作レベルのVHDLを使うと、デジタルシステムの最も複雑な応用の一つであるCPUも簡単に記述することができる。しかし、CPUが動作レベルでモデル化できたとしても、ICチップとして作り上げるまでには、この後いくつかの工程を経る必要がある。そこで、動作レベルのVHDL記述が、それらの作業の中でどのような位置を占め、どのような役割を担っているかを述べる。図2に典型的な開発フローを示す。

(1) 仕様決定

まず、どのようなシステムを作るかを決める。例えばCPUの場合には、アーキテクチャの決定と命令セットの決定、メモリの構造などがこの段階で決まる。

(2) 動作レベル設計

仕様が定まった後、これに基づいて動作レベルの設計を行う。作業は動作レベルでCPUを記述し、シミュレーションを行う。このレベルでは、プログラムの動作試験が可能であり、実際にプログラムを組み、実行させることもできる。プログラミングとその実行を通して、アーキテクチャの不具合、命令セットの問題点等の情報を得ることができ、この段階で仕様が固定したなら、ソフトウェアの開発をスタートすることまで可能となる。問題が起こった時には、設計の変更を行う。問題によっては仕様変更まで必要となる。

(3) RTL設計

動作モデルで完成したCPUを、順序回路設計の手法により、レジスタと組み合わせ回路で構成する、RTL設計(Register Transfer Level)を行う。組み合わせ回路でデータを処理し、それをレジスタで蓄えるレベルで設計する。組み合わせ回路は未だ、ゲートを意識する必要はなく論理式で扱ってよい。

RTL設計が終ると動作レベルでの結果と対比させ、正当性を検証する。検証は主にレジスタ間のデータ転送のタイミング検証になる。この段階で不都合が見つかった時、適時設計変更を行う。

(4) 論理合成とゲートレベル設計

RTLレベル設計が完成すると、次はこれを元に論理合成を行い、ゲートレベルの設計を行う。組み合わせ回路がゲートの組合せとして構成された、構造レベルの記述になる。この段階で論理シミュレーションによって、正当性の検証を行う。ゲートプリミティブとして何を用いるかは、重要である。

MOSトランジスタレベルでゲートを設計することも、論理ファミリーICで構成することも可能である。CPUをチップ化することを考えると、最終的にはMOSトランジスタを用いることになるが、小規模ならば検証用のボードシステムとして、論理ゲートICで試作を行うこともできる。

(5) レイアウト設計

最後の段階はIC設計になり、シリコンチップ上にどのようにトランジスタ等を、レイアウトするかのレイアウト設計になる。ここで作ったマスクパターンによって、チップが完成する。FPGAを利用する場合はFPGAへの配置配線データの書き込み作業がレイアウト設計に相当し、市販のICへの書き込みのみで実現可能である。

各設計工程の終りには、検証と必要に応じて仕様変更再設計が行われる。検証にはテスト信号が必要となるが、この作成もHDL設計では重要である。段階をおって設計を進めることができるVHDLでは、このテスト信号の作成を容易に行える。

例えば、動作モデルの段階で、実際にプログラミングの動作が行えることもその一つで、このレベルではプログラムそれ自身が、一つのテスト信号と見なすことができる。

検証と設計変更で大切なことは、変更をできるだけ前の段階へ戻さないことである。変更が遡るレベルが深くなればなるほど、設計変更の影響が大きくなり、作業が難しくなる。言い換えると、検証は各レベルで徹底的に行い、早い段階で問題点を見つけることが重要となる。この点からも動作レベル、データフローレベル、構造レベルを各レベルでモデル化して、シミュレーション化できるVHDLの存在価値は大きい。

また、VHDLでは、各レベルを混在してシミュレーションができる点も、検証を容易にしている。機能部分を分割して設計するとき、設計している部分のみのモデルを置き換え、他は確実に動作する部分を用いることにより、問題箇所を限定でき、検証を容易にする。

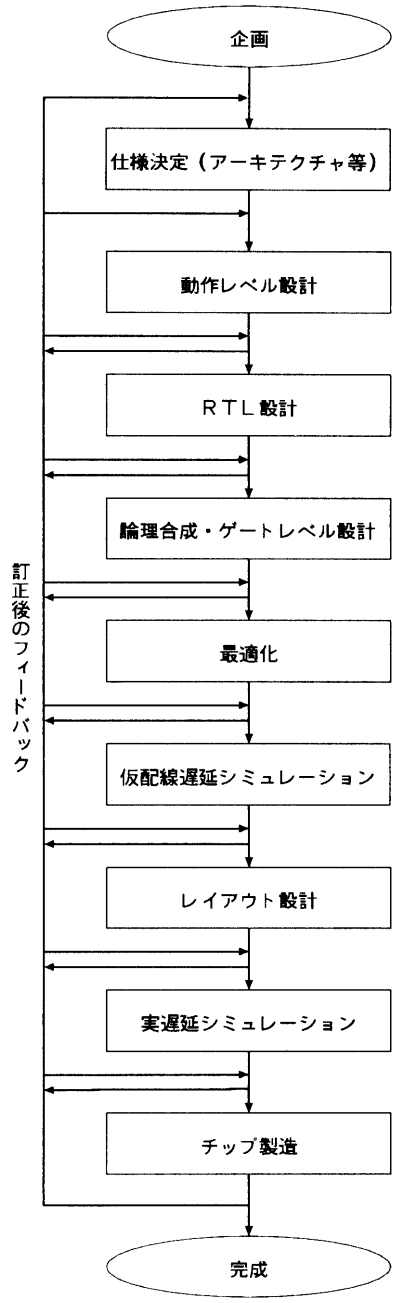

図 2: VHDL による開発作業の流れ

3. ALU 設計仕様

3.1. 基本構成

ALU の入出力仕様は以下の通りである。尚、内部構成は図 3に示す。

1. 入力仕様

X - 被演算信号 (8 ビットバス)
Y - 演算信号 (8 ビットバス)
Control - 制御信号 (4 ビットバス)

2. 出力仕様

Z - 結果信号 (8 ビットバス)
Flag - フラグ信号 (4 ビットバス)

- 被演算信号、演算信号、結果信号は負の数を 2 の補数で表す。
- フラグ出力は"Flag"の 4 ビットのバスで、キャリーフラグ、オーバーフローフラグ、ゼロフラグ、符号フラグを出力する。フラグが検出されたときは 1、されないときは 0 を出力する。

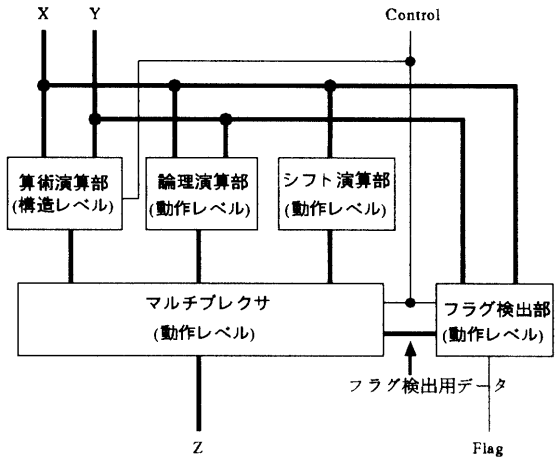

図 3: ALU 内部構成

3.2. 命令セット

設計する ALU は表 1の 16 種類の演算命令を処理する。命令番号の値を Control に入力することで、これらの演算は実行される。

表 1: 演算命令の一覧

命令名	命令番号	演算内容
無命令(IDLE)	0x0	何もしない
加算命令(ADD)	0x1	Z←X + Y
減算命令(SUB)	0x2	Z←X − Y
乗算命令(MUL)	0x3	Z←X * Y
インクリメント命令(INC)	0x4	Z←X + 1
ディクリメント命令(DEC)	0x5	Z←X − 1
論理積命令(AND)	0x6	Z←X ∧ Y
否定命令(NOT)	0x7	Z←¬X
論理和命令(OR)	0x8	Z←X ∨ Y
排他的論理和命令(XOR)	0x9	Z←X ⊕ Y
左論理シフト命令(LLS)	0xa	Z←Xを左に1ビット論理シフトした値
右論理シフト命令(RLS)	0xb	Z←Xを右に1ビット論理シフトした値
左算術シフト命令(LAS)	0xc	Z←Xを左に1ビット算術シフトした値
右算術シフト命令(RAS)	0xd	Z←Xを右に1ビット算術シフトした値
左回転命令(LRO)	0xe	Z←Xを左に1ビット回転した値
右回転命令(RRO)	0xf	Z←Xを右に1ビット回転した値

4. 設計及び試作

4.1. 設計環境

ALU を試作し，動作確認するために FPGA に回路を実装した．ALU の VHDL 設計と回路実装に必要な環境と手順は以下の通りである

(1) 使用機器
- パーソナルコンピュータ
  32 ビット PC-AT 互換機
  (メモリ 64MB
  OS：Windows Ver3.1)
- 実装用ボード
  KITE マイクロプロセッサボード
  (XILINX-FPGA：XC4010 搭載)

(2) 設計用ソフトウェア
- VHDL コンパイラ
  (ViewSynthesis Ver5.0.2x)
- シミュレータ 及び FPGA ダウンロード
  (XACT STEP Ver5.2.1)

4.2. 実装環境

実装に利用した KITE マイクロプロセッサボードは、搭載してある FPGA にユーザの設計した応用回路を書き込む (PC からダウンロード) ことによって、自製のハードウェアを構築できるものである。また、ボード上に各種の入出力ピンやスイッチ、LED ランプが用意されており、所望の入力データを設定したり、内部動作の観測ができる。ボード上の主な構成要素は以下の通りである。

(1)CPU：XILINX － FPGA XC4010
(ゲート規模 10K、論理ブロック数 400、
ピン数 225、 RAM 内蔵)

(2)RAM：128K バイト
(デュアルポート構成)

(3)ROM：64K バイト
(サンプルプログラム書き込み済み)

(4)CONFIG ROM：64K バイト
(KITE-1/KITE-2 マイクロプロセッサ
構成データ書き込み済み)

(5) 汎用入力ポート：8 ビット トグルスイッチ

(6)I/O ポート
入力ポート：16 ビット トグルスイッチ
出力ポート：16 ビット LED ディスプレイ
拡張入出力ポート：60 ピンコネクタ

(7) ダウンロード専用コネクタ：
(FPGA の回路構成データを PC から
ダウンロードするためのもの)

(8) 観測用表示器
LED ディスプレイ：28 個
LED ランプ ：26 個

以上の構成要素の内、本稿の試作で利用する部分は、CPU、汎用入力ポート、I/O ポート、ダウンロード専用コネクタ、観測用表示器である。

尚、CONFIG ROM に書き込まれているデータは、これを FPGA にロードすることにより、KITE － 1 マイクロプロセッサボード、あるいは、KITE － 2 マイクロプロセッサボードとして使用できる。計算機教育/演習用に利用することを目的として設定されているものであるが、今回は自製の ALU を設計することが目的であるので利用しない。

4.3. 設計と実装の手順

今回の VHDL を用いた設計から KITE ボードへの実装までの作業手順を示す。

(1) 回路のソースプログラムの作成

VHDL による回路記述のソースプログラムはテキスト形式であるので、適当なテキストエディタを用いて VHDL のソースを作成する。図 3 に示すように、ALU の算術演算部は構造モデルで設計し他のモジュールは動作モデルで設計した。各モジュールの詳細と VHDL 記述については文献 4)。
尚、最近では回路機能 (状態遷移など) をグラフィック形式で表現し、これを VHDL 入力とする方法も利用開始されている。

(2) VHDL コンパイルと回路の合成

ViewSynthesis を使用して、作成した VHDL の回路ソースプログラムを回路図へ変換する。ゲートレベルの回路図が得られる。

(3) FPGA の回路構成データの作成

XACT-Step を使用して、FPGA 上で回路を動作させるため、回路図を FPGA 専用の回路構成データに変換する。

(4) ユーティリティ回路の作成

実装後の回路動作を目視で確認するために KITE ボードの汎用入力と I/O ポートを利用するが、このために必要な回路も FPGA で実現する。ALU への命令とデータ入力は汎用入力スイッチと I/O ポートによる。I/O ポートを使用するためには、ポートに対して入力要求とアドレス指定が必要

であり、それらの回路設計が必要となった。また、出力にはLEDディスプレイとLEDランプを用いる。ディスプレイを動作させるドライバはボードに搭載済みであるので、特別な処理はなくても利用できた。以上の入出力を実現するため、FPGAチップとKITEボードを接続しているピンを考慮して、ボード上の配線を行った。

(5) FPGAへ回路構成データをダウンロード

XACT-Stepを使用して、FPGAへ回路構成データをダウンロードする。完了次第、回路は動作開始可能である。

4.4. 結果と評価

VHDLで回路設計後、XACT-Stepのシミュレータにより，動作シミュレーションを行った。図1の仮配線遅延シミュレーションに該当するものである。図4はそのシミュレーション結果の一部である。遅延が大きいのはフラグ検出の部分であった。中でも、乗算命令の出力結果を受けて検出されるフラグの遅延が最大であった。本例の場合 (図4)、遅延時間は 2.1 ns (1 ns = 10E-9 sec) であった。この逆数 (1/2.1 ns = 470 * 10E6) から、設計したALU単体のの理論的な動作速度が見積もられる。ただし、実際のMPU全体の動作速度は、命令解読、データ転送、結果出力など、他のプロセッサ内の処理工程が大きいので、ALU単体の動作速度よりははるかに遅くなることに注意しておく必要がある。

FPGAの使用率については、XC4010の使用可能な論理ブロック数 400 の内、今回の設計で占有されたのは155ブロックであり、38%の使用率であった。最終的に、KITEボードで操作し、演算命令を逐次実行し、正常に動作していることを確認した。

5. まとめ

本稿では、論理システムの設計に関して、従来の回路図入力による設計方法に代えて、ハードウェア記述言語VHDLとCADツールを用いた設計方法を試行した。また、設計結果を実機テストするための簡便な環境としてFPGA搭載のKITEボードを利用し、試作環境として使いやすいものであることを確認した。

今回の設計対象はALUだけであり、規模は小さいが、VHDL設計、シミュレーション、FPGA実装、実機テスト といった工程で、上流の設計からハードウェア試作までの一貫した処理が、比較的小規模・廉価な設備やCADソフトで実行できたことから研究室レベルでの試作環境として十分に有効であることを示している。今後は、この試作環境を利用して、より規模の大きいディジタルシステムの設計に応用すること、また、近年、問題になっている論理システムの消費電力や動作速度の問題などに対処できる高性能システムの設計・試作について検討することなどを予定している。

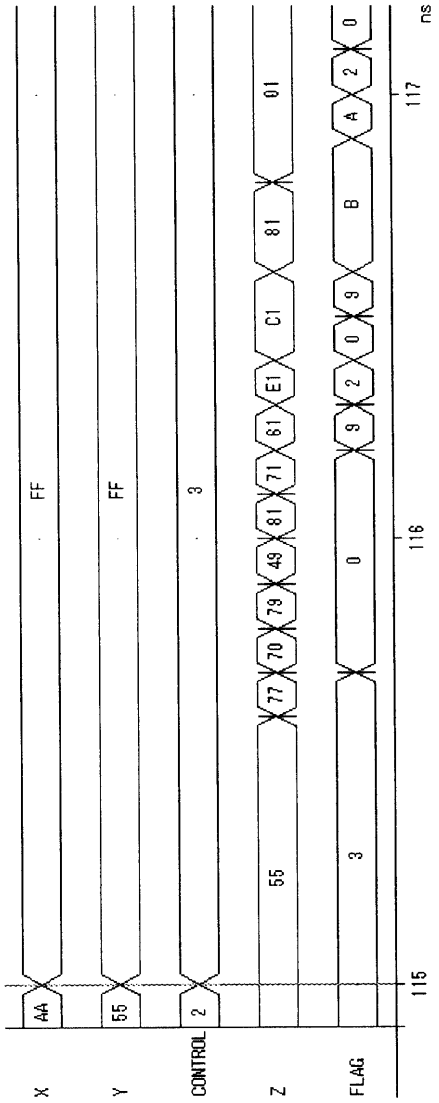

図 4: ALUのシミュレーション結果

参考文献

1) Z.Navabi (佐藤訳): VHDLの基礎, 日経BP社, 1994.

2) たとえば, 長谷川裕恭: VHDLによるハードウェア設計入門, CQ出版社, 1997.

3) KITEプロジェクト編: KITEマイクロプロセッサボード PLUS+ 取扱説明書, 1995.

4) 川島由治: VHDLによるALU設計とKITEボード応用, 平成9年度山口大学工学部知能情報システム工学科卒業研究論文, 1998(2月).

(1998.5.15 受理)

ALU DESIGN USING A HARDWARE DESCRIPTION LANGUAGE VHDL AND IMPLEMENTATION ON AN FPGA BOARD

Yuji KAWASHIMA, Takahiro WATANABE, Yoshihiro TANADA and Kazuo MORIKAWA

As the integration and the design complexity of digital systems are increasing, a sophisticated design style like a hierarchical design and many efficient CAD tools are widely used. Especially, design using a hardware-description-language (HDL) and a logic synthesis is most interesting. We try to design a small-sized digital system (ALU) by VHDL and a synthesis tool, and implement it using an FPGA board.