Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya

Reducing energy with asynchronous circuits

Author:

Daniel Rivas Barragan

Supervisor:

Jordi Cortadella Fortuny

Dept. de Llenguatges i Sistemes Informàtics

DADES DEL PROJECTE

Títol del Projecte: Reducing energy with asynchronous circuits.

*Nom de l'estudiant:*Daniel Rivas Barragan

Titulació: Enginyeria Informàtica

Crèdits: 37.5

Director/Ponent: Jordi Cortadella Fortuny

Departament: Llenguatges i Sistemes Informàtics

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Roger Espasa Sans

Vocal: Josep Elgueta Monto

Secretari: Jordi Cortadella Fortuny

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva:

Data:

Contents

List of Figures

List of Tables

Preface

In recent years, energy consumption is taking a lot of importance in the field of electronics. Nowadays, this is becoming even a critical issue because of the growing smartphone market or low-voltage solutions, for example. Accordingly, every time more efforts are devoted to reduce energy consumption in electronic devices to make them more efficient or to prolong their battery life, among others. This implies that the natural improvements given by the technology are not enough: more overwhelming solutions are needed.

Reduction of energy consumption can be achieved in so many ways. Sometimes it comes from an improvement in the technology used for manufacturing semiconductors and others from using a better and more efficient design. Nevertheless, it has not been carried out any really important change in this field.

Different solutions have been presented but, rather than specific design modifications, they usually imply a change on the design paradigm and on the design techniques. Note that this is a huge industry and needs a strong argument and big improvements before even thinking on changing the whole industry methodology. Thereby, it is reluctant to change its working paradigms and methods and it prefers to keep using more conservative solutions, partly because of the leak of automated tools for new design paradigms but also partly because of just habit.

This project presents an implementation of the elastic clocks [9], which is a method for implementing asynchronous circuits [4], that tries to get the best of the synchronous paradigm but taking advantage of the benefits of asynchronous circuits. This is used for reducing energy consumption while variability, enemy number one in synchronous circuits, is mitigated as far as possible. This means that a reduction on power consumption is achieved at the time some problems of the synchronous circuits are being solved.

Moreover, the idea of a closed-feedback loop is used in combination with the asynchronous circuits to take full advantage of both techniques. A closed-feedback loop in which the output is monitorized and feeds the data to a controller which adjusts some parameters as necessary to maintain the desired system output. This is used for the workload and control the frequency in order to adjust it at its optimum value, thus, it will be just as fast as needed at any time. This is done reducing voltage and, hence, reducing the power consumption.

Organization

The document is divided in three parts. The first part exposes what is called the state of the art. In this part all the theoretical background needed to understand the project is explained. First of all, in Chapter 1 the circuit design flow is explained in order to have a global understanding on how a chip is made - from high-level language until a valid layout -. Then, a brief explanation about how chips are manufactured is presented in Chapter 2 with the purpose of knowing what can provoke variability, which is one of the main reasons to use asynchronous circuits. Finally, in Chapter 3 different approaches and methods of implementing asynchronous circuits are analyzed and the reasons of choosing elastic circuits to be implemented are exposed.

During the second part introduces, in Chapter 4 the elastic clocks implementation is presented along with the closed-feedback loop. During Chapter 5 the elastic clock is tested, using a test CPU, introducing noise and scaling the voltage in order to see how it supports a more aggressive environment than its synchronous counterpart while reducing the power consumption. Finally, in Chapter 6 a real case of use of the closed-feedback loop is presented. it comprises an *Infinite Impulse Response* system, which works with the data provided by an *Analog to Digital Conversor.*

Within the third part, in Chapter 7 the schedule of the project and the economic feasibility of the project are presented and described. To conclude, Chapter 8 analyzes the final state of the project and the achieved aims that have been proposed earlier in this chapter and some possible improvements are commented in form of future work.

As an appendix, the development tools are briefly explained and the RS-232 protocol, needed for data transmission between the board and the power supply and the computer, is explained along with some implementation explanations without introducing implementation details.

Chapter 1

Circuit design flow: from specification to layout

During this chapter will be presented the different steps that comprises the development of an integrated circuit, , from the initial idea until it is ready to be sent to the factories. Manufacturing process will be presented in Chapter 2.

The process of designing a chip is very complex and requires up to several months – even years in more complex designs – from the initial idea to a completely functional device. With the purpose of putting some order in the process there are some stages defined.

Figure 1.1 shows the standard design flow, which comprises from the initial specification to the chip manufacturing or FPGA programming in case of working in an FPGA environment. Figure 1.1 also shows how some stages are iterative once is completed, a timing analysis and other constraint analysis need to be done in order to know at each stage that no constraints are violated.

1.1 Specification

The first thing to do is the specification. Before start programming or designing we need to define several properties.

Specification defines not only the main operations and functionalities of the chip but also other design factors like chip area, power consumption, speed or cost. It is vital to keep all this factors in mind in every step of the design flow because any

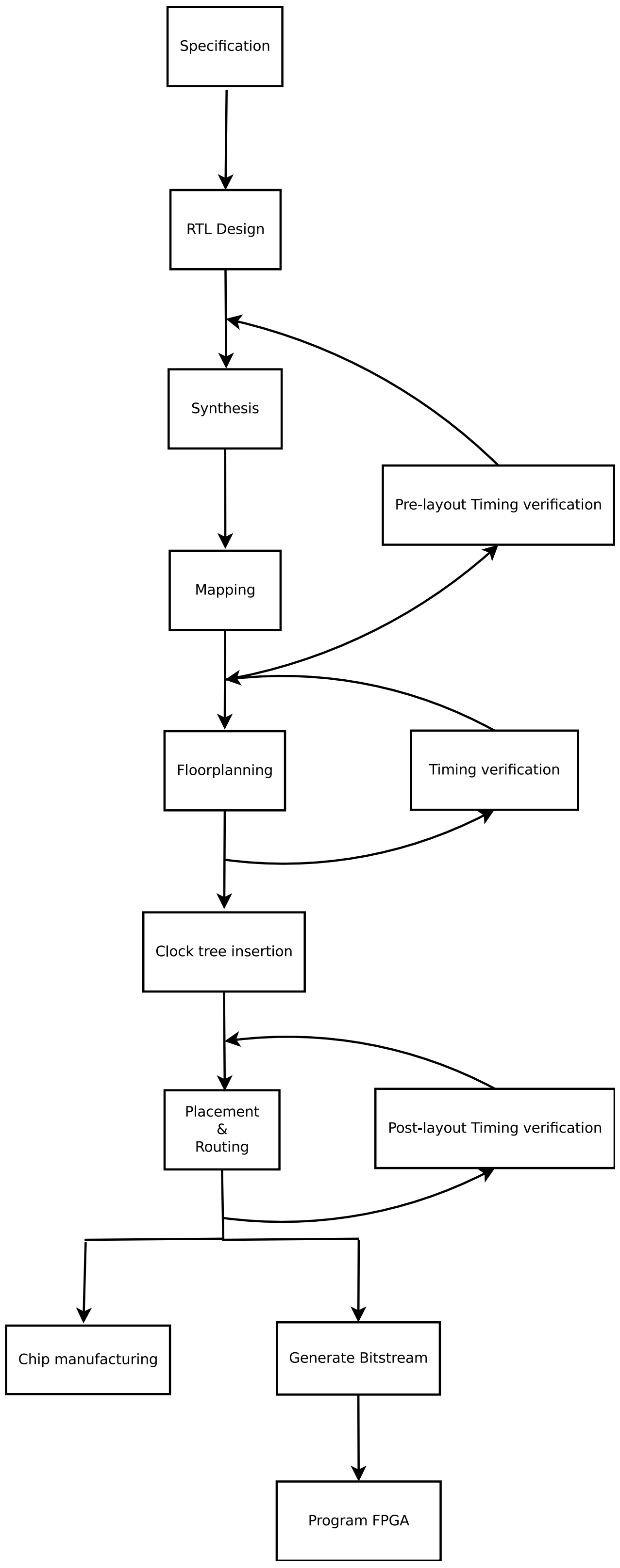


Figure 1.1: IC Design flow.

change could mean months of delay in the release date.

Different groups usually carry out each step of the design flow. For example, one group implements in HDL while another one enforces the timing constraints and if the second group finds out the first did not implement the design properly the project has to go backwards and has to be redone whilst the second waits. The circuit design flow is very dependent between stages even being encapsulated.

1.2 Hardware Description Language

HDL stands for Hardware Description Language. In electronics is any language for formal description and design of electronic circuits. HDL syntax and semantics has notation for expressing concurrency.

HDLs, such as Verilog, differ from software programming languages because they include the notion of time and ways of describing the propagation of time and signal dependencies (sensitivity). Time is used for simulation or timing analysis, for example.

The most commonly used languages are Verilog and VHDL (VHSIC Hardware Description Language). Our project is developed using Verilog even though there are some parts in VHDL. The differences between them are mainly syntactical.

1.2.1 Verilog

Verilog[1] is a hardware description language used to model electronic systems. Used in the design, verification and implementation of digital logic chips.
A Verilog design consists of a hierarchy of modules. These modules perform operations and are connected with other modules through a set of input and output ports. Verilog is used at the register-transfer level of abstraction (RTL).

Histoy Verilog was the first modern hardware description language to be invented (1984). Verilog represented a tremendous productivity improvement for circuit designers and it allowed the emergence of EDA software, which made possible the great

[1]For more information and tutorials on Verilog than is provided in this section refer to [1]

silicon industry that exists today. Cadence Design System - EDA and engineering services company - purchased Verilog in 1990.
Verilog quickly became very popular but arose VHDL as a strong competitor. Cadence decided to make the language available for open standardization. Verilog was submitted to IEEE and became IEEE Standard 1364-1995, more known as Verilog-95.
Verilog was the first modern hardware description language to be invented (1984). Verilog represented a tremendous productivity improvement for circuit designers and it allowed the emergence of EDA software, which made possible the great silicon industry that exists today. Cadence Design System - EDA and engineering services company - purchased Verilog in 1990.
Verilog quickly became very popular but arose VHDL as a strong competitor. Cadence decided to make the language available for open standardization. Verilog was submitted to IEEE and became IEEE Standard 1364-1995, more known as Verilog-95.

Syntax There are a few important things to know about the Verilog syntax in order to understand how it works.

Verilog allows defining combinational logic and sequential logic. Combinational logic can be defined using typical Boolean expressions. The portion of Verilog code shown in table 1.1 is an example of a combinational block:

```
module combinational(a,b,c,y);
  input a,b,c;
  output y;

  wire y;

  assign y = (a & b) ^ (b & ~c);
```

Table 1.1: Verilog combinational block.

The evaluation priority is first parenthesis and then logic operations. The code above would be translated to the circuit shown in Figure 1.2:

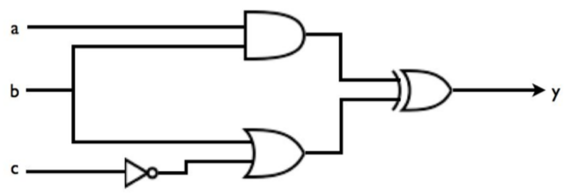


Figure 1.2: Combinational circuit.

Combinational variables are declared as wires while sequential as regs. Declaring a register does not mean that in the final circuit it will still be a register. Since optimizations are done, a variable declared as a register can be implemented with a single wire if necessary.

Also, there is more than one way of working with sequential logic.

Blocking and nonblocking statements There are two assignment operator, a blocking assignment '=' and a non-blocking assignment '<='. The non-blocking assignment is one of the aspects that differ HDLs to normal procedural languages. The following code shows an example of its usage:

While we are using a non-blocking assignment, the order is irrelevant and *flop1* and *flop2* will swap values every clock. In the other hand, when using the = assignment the value is updated immediately. If we use the blocking assignment = then *flop1* <= *flop2* will be executed in first place because *flop1* and values would not have been swapped. Actually, as in traditional programming, *flop1* would simply take the value of *flop2* and the second statement would be removed for being redundant.

Always clause The *always* clause defines a block that is executed, as its name indicates, always. The execution will enter the always block depending on the *sensitivity list* which indicates the signals that will be monitored. For example code in table 1.3:

It will be executed whenever signal a or b changes. The *always* clause also allows to define trigger options like *posedge* or *negedge* which will make the clause to be executed at a positive edge of the signal or the negative edge of the signal, respectively.

```
module toplevel(clock, reset);
  input clock;
  input reset;

  reg flop1;
  reg flop2;

  always @ (posedge reset or posedge clock)
    if (reset)
    begin
      flop1 <= 0;
      flop2 <= 1;
    end
    else
    begin
      flop1 <= flop2;
      flop2 <= flop1;
    end
endmodule
```

Table 1.2: Example of nonblocking statement.

```
always @(a or b)
```

Table 1.3: Verilog always clause example.

Signal types Signals that are driven from within a process (an initial or always block) must be type *reg* and signals that are driven from outside a process must be of type *wire*. Defining a signal as *reg* does not necessarily imply a hardware register, this is decided by the synthesizer.

It is important to notice that FPGA tools allow initial blocks where *reg* values are established instead of using a 'reset" signal for the same purpose. This is possible because an FPGA' initial state is downloaded into the memory tables of the FPGA. For this reason, ASIC synthesis tools do not allow this, because an ASIC is an actual hardware implementation.

Four-valued logic The standard defines a four-valued logic with four states: 0, 1, Z (high impedance) and X (unknown logic value).

Constants definition The syntax for defining constants is: *<Width in bits>'<base letter><value>* Base letter can be h (Hexadecimal), d (Decimal), o (Octal) or b (Binary). When width is smaller than value, then leftmost bits of value are truncated. When width is larger than value, then leftmost bits are filled, based on the value of the leftmost bit in value '0' or '1' are filled with '0', 'Z' are filled with 'Z' and 'X' are filled with 'X'.

1.3 Synthesis

Verilog makes the programmer work easier. However, not all Verilog statements have a direct translation to hardware. For example, an AND can be directly translated to an AND gate in hardware but there is no CASE gate and if a case statement is used, it will need to be translated first to what is called *synthesizable Verilog.* Synthesizable Verilog is a subset of statements that have direct translation to real hardware. The process of translating abstract Verilog to synthesizable Verilog is called *synthesis.*

Synthesis software algorithmically transforms the Verilog source into a netlist, a logically equivalent description consisting only of elementary logic primitives (AND, OR, NOT, flip-flops, etc.) that are available in a specific FPGA or VLSI technology. A netlist describes the connectivity of an electronic design. It provides information of how modules are related between them and which pins and ports are connected.

With all of this, we can say that logic synthesis is a process by which an abstract form of circuit behaviour (typically specified in RTL) is turned into a design implementation in terms of logic gates. This process is analogous to a software compiler converting a high-level C program into a processor-dependent assembly language code. The synthesizer ensures mathematical equivalency between the synthesized netlist and original RTL description.

1.3.1 Logic minimization

One of the most important things a synthesizer does is to minimize the logic. Logic minimization is directly correlated to area minimization and is one of the optimization steps. Boolean logic minimization started with Karnaugh maps but it was not until the *Quine-McCluskey* algorithm was introduced that logic minimization became fully automated. However, although *Quine-McCluskey* algorithm became very important during the first ages of EDA, since it is a two-level minimization its importance in very-large-scale integration (VLSI) is quite reduced because most designs use multiple levels of logic. Also it is far from efficient in terms of processing time and memory because the truth table length increases exponentially with the number of variables.

In the 2-level minimization field the Espresso heuristic logic minimizer (Robert K. Brayton, Berkeley) has become the standard for this operation. It has been incorporated as a standard logic function minimization step into almost any contemporary logic synthesis tool even though it is used mainly for academic purposes.

The Espresso algorithm is a heuristic method, thus the minimization result is not guaranteed to be the global minimum but in practice it is very approximated and the solution is always free from redundancy.

Multi-level logic minimization The synthesis tool starts from the RTL description of the desired design and constructs a multilevel Boolean network.

Then, several optimizations are made to this network. First of all are applied the technology-independent techniques, like applying a given minimizer algorithm in order to reduce the number of literals.

Finally, the result of applying the technology-independent optimizations is taken and are applied the technology-dependent optimizations, which transforms the circuit into a network of gates – netlist – in a given technology.

1.4 Timing analysis

After the synthesis is finished it is very important to perform a timing analysis in order to know that the timing constraints are not being violated. After the synthesis there are still a lot of unknown information about how the circuit is going to finally behave: different placements, routings or mappings can make vary widely the behaviour of the final circuitry. Hence, the timing analysis will be incomplete at this point but it would give an approximation and, at least, would give the information of violated constraints which is enough reason to stop the process and try again with more aggressive optimizations or scarifying other parameters with more margin – for example, if it is still possible taking into account initial specification, consume more in order to reduce critical path – . Sometimes it becomes impossible to satisfy all the timing constraints due to inefficient RTL or very restrictive initial specification. In this case, the design flow needs to go backwards at some point where these things can be rewrite.

Timing analysis is not a unique step and every time new decisions are made and new information is obtained a new timing analysis needs to be performed in order to ensure decisions were correct.

The result of a timing analysis is a critical path and, hence, the maximum speed of the circuit. The speed determined by the timing analysis is always a safe speed at which the circuit can run without failures working in a normal environment–the working environment is defined before the timing analysis is performed in terms of input voltage range or temperature range –.

1.5 Mapping

Technology mapping is the phase of logic synthesis when gates are selected from a technology library to implement the circuit. The gates contained in a *technology library* are called *standard cells.*

1.5.1 Standard Cells

Standard cell methodology is an abstraction method in the design flow. The layout is all encapsulated into an abstract logic representation (such as AND gates). This allows the designer to focus in high-level implementation while another focuses on the hardware implementation.

A standard cell usually provide the most basic boolean logic functions – like AND, OR, XOR, XNOR, inverters – and storage elements – flipflop or latch –. Even though, sometimes more complex cells can be provided.

A standard cell performs a logic function and knowing it is enough information for the high-level designer at the abstraction level – it is called the logical view of the cell – but more information is needed when running further steps in the design flow. At the manufacturing step, there will be needed what is called the physical view which determines how the cell is made.

1.5.2 Technology libraries

A standard cell library is a collection of standard cells descriptions. Standard cells are designed with a fixed height but variable width. This enables them to be placed in a row making easier the automated process.

In addition, even though a 2-input NAND or NOR function is sufficient to form any boolean function set, a standard cell library usually contains multiple implementations of the same logic function.

This variety allows trade-offs in terms of area, speed and power consumption during synthesis, placement and routing. For example, if the final circuit appears to be smaller than expected some cells can be changed for bigger and faster cells in order to reduce critical path – if power constrains allow it –.

Thus, the automated tools need information to decide which standard cell to choose. This information is contained in the libraries. A standard cell library contains mainly two types of information:

1. Library Database: Contains reduced information about layout or schematics but enough for the Placement and Route tools.

2. Timing abstract: provides functional definitions, timing, power and noise information for each cell.

Technology libraries are developed and provided by the foundry operator because the resulting library depends on the technology available in the foundry – for example, the size of the transistors –.

After the mapping, delays and power consumption of the cells that will be in the final design are known. For this reason, a new timing analysis is performed with new data. If it fails, the mapping needs to be redone.

1.6 Placement and Routing

Place and route is the final step in the design flow where, as implied by the name, the components are finally placed in the board and then interconnected but first, there is an intermediate step: floorplanning.

A floorplan of an integrated circuit is a schematic representation of tentative placement of its major functional blocks. Usually chip area is therefore in some cases given a minimum area in order to fit in the required number of pads, which can give an approximation of the final chip area. After this step, there is a new timing analysis and then, placement and routing can be performed.

Placing and routing is done in two steps. Placing the components first, then routing the connections between the recently placed components. Typical placement objectives are:

- Minimizing total wire length. This helps minimize chip size, cost and also power and delay, which are proportional to wire length.
- Timing: The placer must ensure that no path exists with a delay over the maximum specified.
- Congestion: A congestion region may result in an impossible route.
- Power minimization: distribute the locations of cell components to reduce overall consumption and mitigate hot spots.

The placer takes the netlist and produces a valid layout. Then, the router takes the result layout of the placer and adds wires needed to properly connect the placed components. The routing problem is NP-complete. Therefore, routers do not attempt to find an optimum result but a good enough solution through heuristics.

The placement is not absolute and if the router finds out the resulting placement is impossible to be routed without violating any restriction the flow can go backwards and the placement repeated.

After the Placement and Routing step we obtain a valid layout. At this point, once we know information about the layout, there is a new and final timing verification in order to know if the final design satisfies all the timing constraints. If it does not satisfy them all, again, the last step needs to be redone but if it does satisfy them, no more steps are needed and the design is completed. At this point, the design can be sent to manufacturing in case of being an IC. If it belongs to an FPGA design, the last step is generate a bitstream, which carries all the information to program the board. However, even the FPGA board, as IC that is, had to pass through all this stages once in order to be what it is. For this reason, the FPGA suffers the same variability problems of variability as others ICs.

Chapter 2

Manufacturing process

During this section, the manufacturing process is briefly presented in order to understand how it can make vary the final result because not every chip achieves the same throughput even coming from the same design and even the same wafer. This is produced by variability, sometimes provoked by errors in the manufacturing process, which is explained later in this chapter.

Once the design is finished and exhaustively verified is time to send it to the fabs, the semiconductor fabrication plants. The fabrication process of integrated circuits is a very complex and multi-step process. Each manufacturer has its own techniques even though they are very similar.

Integrated devices are usually made of silicon even though in some specialized applications can be used various compound semiconductors. The entire manufacturing process, from sand to packaged chips, takes up to several months.

2.1 Silicon wafers

The first step in the manufacturing process is to create the silicon wafers where ICs will be printed.

A typical wafer is made out of extremely pure silicon (99.9999999% purity) and measured by mass, silicon makes up 27.7% of the Earth' crust and is the second most abundant element in the crust, with only oxygen having a greater abundance. However, the way silicon is found in the nature is not suitable for manufacturing uses. Silicon is found in a compound known as silicate (SiO44-), which contains always silicon and oxygen and, sometimes, is found along with other elements like aluminium, magnesium or calcium.

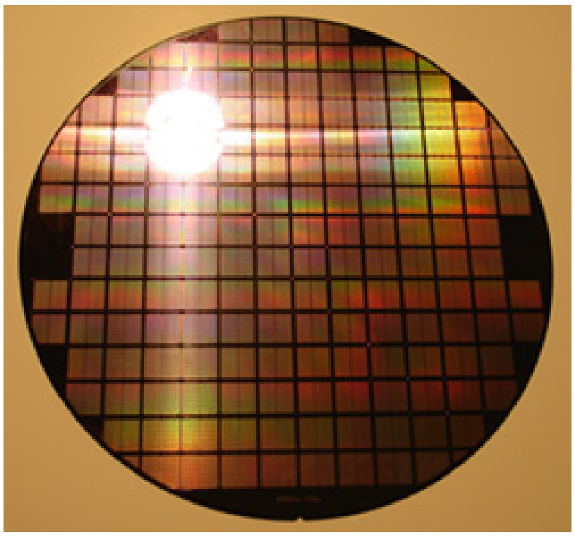


Figure 2.1: Silicon wafer.

Years ago, when transistor' size was of the order of micrometres, silicon purity was not the issue that is today. Impurities can produce undesirable changes in silicon properties such as conductivity.

As most of the elements found in the nature, silicon is not found 100% pure. In order to achieve an acceptable purity, silicon is purged in an argon atmosphere at over 2500 °F to remove oxygen or nitrogen, which produce impurities.

One process for forming crystalline wafers is known as Czachralski growth. In this process, a cylindrical ingot of high purity monocrystalline silicon is obtained of over 200Kg with a diameter up to 30cm, which will be cut in wafers of 400um to 600um thick.

Monocrystalline silicon consists of silicon in which the crystal lattice of the entire solid is continuous and unbroken, with no grain boundaries. Having no grain boundaries provides monocrystals with unique properties, particularly electrical.

Wafers are grown from crystal having a regular crystal structure, with silicon having a diamond cubic structure. The structure is anisotropic, which means that some properties like mechanical, optical or electrical depend on the crystal orientation. Ion implantation depths in later steps depend on the wafer' crystal orientation, for example.

This and next steps are performed in dust free cleanrooms to avoid the appearance of any kind of impurities. Cleaning is essential for the wafers as particles lurk at every stage in the manufacturing process.

2.2 Processing

After the wafer is created, from three hundred to four hundred steps are required to obtain a final chip. Following, some of the most important are described.

2.2.1 Oxidation

First, the silicon wafer is oxidized through thermal oxidation in order to obtain silicon dioxide, used as an electrical insulator. Thermal oxidation can be performed through wet or dry oxidation. Wet oxidation is preferred because is faster – slowness of dry oxidation makes it impractical – but yields a lower-density oxide, with lower dielectric strength. Usually a mixed technique is used. The silicon oxide lands over the wafer as a new dielectric layer.

2.2.2 Photoresist

Once we have the new dielectric layer over the wafer, another a layer is created. Wafer is spin coated with a photoresist. A photoresist is a light-sensitive material and can positive or negative, which determines the response to the light. For example, if the layer is formed of positive photoresist, the portion of the photoresist that is exposed to the light becomes soluble to the photoresist developer, a chemical fluid that removes the photoresist, and the unexposed portion remains insoluble to the photoresist developer.

2.2.3 Photolithography and etching

After the photoresist is over the wafer, it is time to use etch the. Photolithographic techniques are used to etch the circuit structure in the wafer. Masks with the opposite structure of the circuit are placed in front of the wafer and UV light is emitted towards the wafer. This makes photoresist soluble in the portions where the transistors will be placed and they are removed.

2.2.4 Ion implantation or doping

Once the circuit is etched on the wafer, it takes place the ion implantation step, where the electrical properties of the silicon will be established. Dopant atoms are shot into the silicon structures. Dopant impurity atoms such as boron or phosphorus can be added to the molten intrinsic silicon in precise amounts in order to dope the silicon, thus changing it into n-type or p-type extrinsic silicon.

Initially, these dopants are distributed unevenly in the silicon lattice. At high temperatures, the dopant atoms become flexible and take over a fixed position in the atomic structure. Now, the photoresist layer is no longer needed and is removed.

2.2.5 Iterating the process

A new layer of silicon is applied over the oxide layer and, once again, a layer of photoresist is applied, exposed and develop.

The process is repeated until the whole pattern is complete in all its layers. From three hundred to four hundred steps are required along with fifteen to twenty different masks with microscopically small patterns.

Figure 2.2 shows the different layers of a transistor.

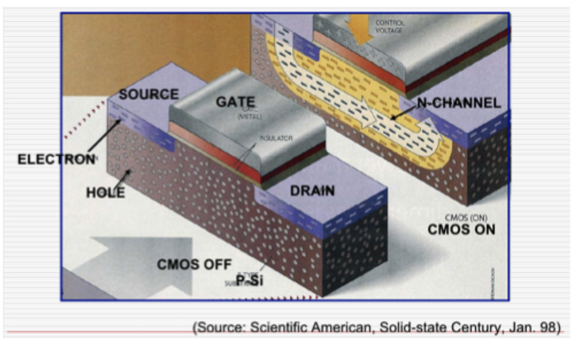


Figure 2.2: Different layers of a transistor.

2.3 Wiring

Copper dominates the next process. Before copper is poured into the trenches for interconnects, a barrier layer is applied. It helps to prevent short-circuits and guarantees reliability. The trenches are then filled with copper. Finally, the excess copper is ground down to the edges of the trenches. This insulates each interconnection from the others.

2.4 Packaging

The final stage in manufacturing process is packaging. With the finest saw blades, the microprocessors are cut from the wafers, bonded to the frames and sealed with a cover. Then, the microchip is ready to be shipped and sold.

2.5 Technology limits

A circuit is manufactured using a given technology and it always has limits, given by the foundry, economical costs or even physical limitations. Moore' law is sometimes used for predict the scalability of the technology and circuits. The most common technology used for constructing circuits is CMOS while the technology used for the transistors is called MOSFET.

2.5.1 Moore' Law

The Moore' law is a rule of thumb in the history of computing hardware made in 1965 by the Intel co-founder Gordon E. Moore. Moore' Law states that the number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years. Figure 2.3 shows the CPU transistors count against its market release dates. Note the logarithmic vertical scale; the line corresponds to exponential growth with transistor count doubling every two years.

Over time, different technologies have been used for combining huge number of transistors into a single chip. During the first years of ICs there was the *small-scale*

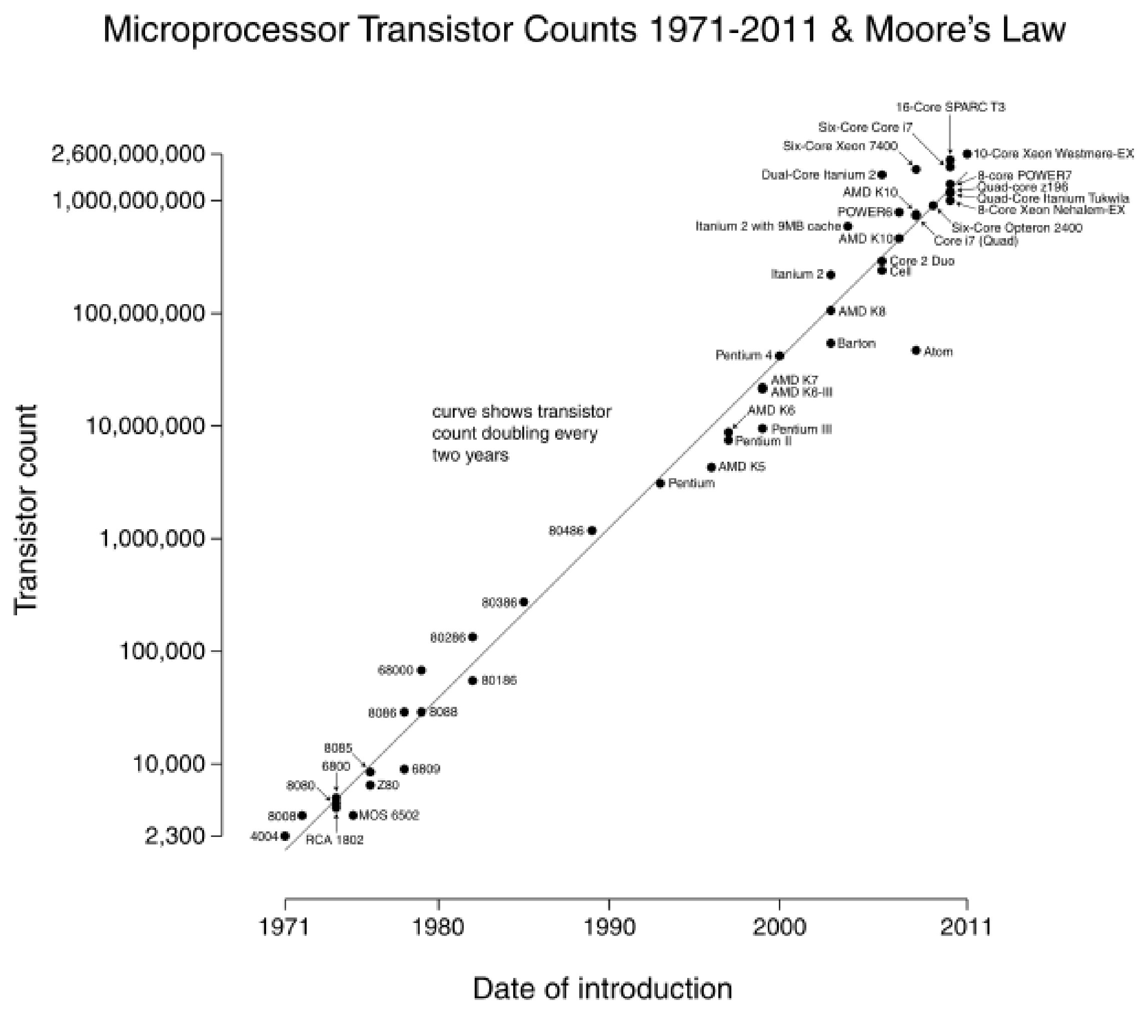


Figure 2.3: Plot of CPU transistors count against dates of introduction. source: en.wikipedia.org/Moore'_law.

integration – SSI – with tens of transistors on a chip and was used in the Apollo guidance computer.

The next step, in late 60', was the *medium-scale integration* – MSI – with hundreds of transistors on a single chip. This started to allow more complex systems. Further development drove, in the mid 70', to the *large-scale integration* – LSI – with tens of thousands of transistors per chip. First microprocessors began to be manufactured.

The final step, starting in the 80' and continuing through the present, was *very large-scale integration* – VLSI –. VLSI started with hundreds of thousands in the early 80' and continues with several billion transistors nowadays.

This level of integrations has been possible due to the shrink of the transistors size. In 1971 the size was 10 μm and 2011 has achieved the mark of 22nm. Table 2.1

shows this progression of the semiconductor's size through years.

Size	Release date
10 μm	1971
3 μm	1975
1,5 μm	1982
1 μm	1985
800 nm (.80 μm)	1989
600 μ (.60 μm)	1994
350 nm (.35 μm)	1995
250 nm (.25 μm)	1998
180 nm (.18 μm)	1999
130 nm (.13 μm)	2000
90 nm	2002
65 nm	2006
45 nm	2008
32 nm	2010
22 nm	2011
16 nm	approx. 2013
11 nm	approx. 2015
6 nm	approx. 2020
4 nm	approx. 2022

Table 2.1: Semiconductor's size through years.

Smaller size implies less area, hence less cost, less power consumption and other benefits. But smaller size also adds more complexity to the manufacturing process because the transistor is made of just a few atoms – 22nm approximately 100 atoms – and impurities become more dangerous to the performance. Moreover, the lithography process becomes more difficult to perform accurately because the UV wavelength exceeds the length of the semiconductor. As we can see in the Figure 2.4, the UV was perfect for lithography until after the year 2000 approximately because it is when the UV wavelength was larger than the transistor's size. This problem is becoming more and more difficult to treat and it is a source of errors in the final result. When the result is not exact or the silicon gets impurities, variability appears.

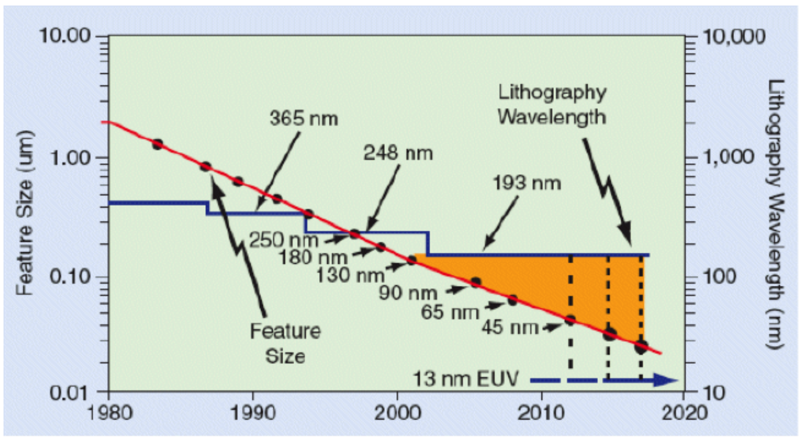


Figure 2.4: Lithography wavelength progression over semiconductor' size.

However, a new technology using EUV – *Extreme UltraViolet* – is expected to be used in the next years, which as a smaller wavelength and it will provide a new way of shrinking the circuits even more without sacrificing reliability.

2.5.2 MOSFET Technology

Metal-oxide-semiconductor field-effect transistor – MOSFET–is a type of transistor used in integrated circuits and a four-terminal device with source (S), gate (G), drain (D) and body (B) terminals. However, the body and the source are often connected.

A voltage drop across the oxide induces a conducting channel between the source and the drain contacts thanks to the field effect. The channel can contain electrons – nMOS – or holes – pMOS –, opposite in type to the substrate – nMOS is made with p-type substrate and pMOS with n-type –. Figure 2.5 shows a MOSFET transistor and its parts.

2.5.3 CMOS Technology

Complementary metal-oxide-semiconductor refers to a technology for constructing integrated circuits. The typical CMOS design uses complementary and symmetrical pairs of p-type and n-type MOSFETs.

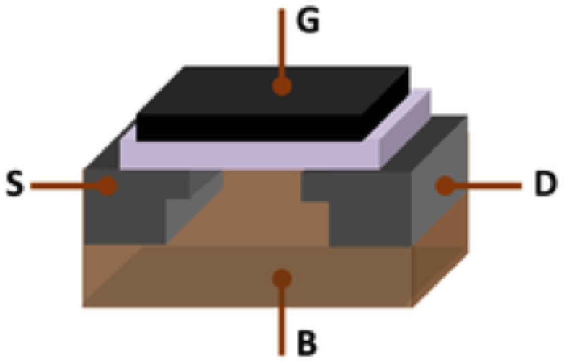


Figure 2.5: MOSFET Transistor.

Thereby, there is only significant power consumption when the transistors are switching between on and off states, hence, in static mode the power consumption is only due to leakage.

Main benefits in CMOS technology are low static power consumption and high noise immunity. Figure 2.6 shows an example of a gate implemented using CMOS technology. In this case it is a NAND gate.

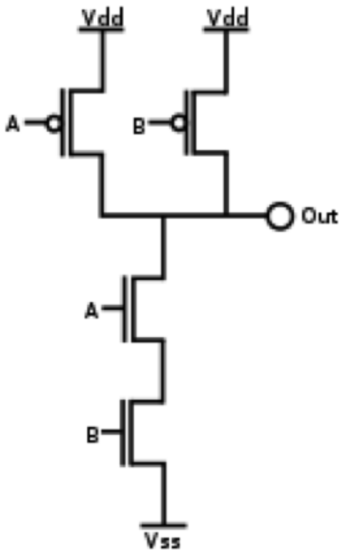


Figure 2.6: NAND CMOS gate.

2.6 Variability

Once we know the theory of how circuits are manufactured, we can understand what can go wrong. Sometimes the etching is not perfect or maybe the amount dopants are not the same at every single point of the wafer. These differences produce what is known as circuit variability, but they are not the only cause of variability.

Variability in circuits comes in different ways and some times in a totally random way. Main cause of variability is what is called PVT. PVT stands for Process-Voltage-Temperature.

2.6.1 Process Variation

This kind of variation is due to deviations during the semiconductor fabrication process. Rather than random, process variation is extremely difficult to determine. This is why usually is treated as a percentage variation in the performance calculation.

Variations can be impurity concentration densities, oxide thicknesses or diffusion depths caused by non-uniform deposition of dopants but mainly because of the limited resolution of the photolithographic process. Lately, with the reduction of the transistor' size, this has become even smaller than the wavelength of the ultraviolet light used in the photolithographic process. The way a transistor works, the width of the gate determines how much current crosses it and the greater the current the greater the speed of the signal to cross and vice versa. Thus, a thinner gate implies a slower gate and the transistors have different transistor lengths throughout the chip – which means different propagation delays throughout the chip –.

Process variations are due to variations in the manufacture conditions such as temperature, pressure and dopant concentrations. ICs are mass-produced, in wafers with dozens of dices in every wafer. Process variations are not only present within a chip but also in different chips in the SoC, in different dices of a wafer and in different wafers. Process variations are very difficult to determine and this is why after the ICs are manufactured, they are tested to prove that even with all variations, the conservative margin added is enough for them to work properly.

After the tests are performed, manufacturers determine if the IC can run at the frequency it is supposed to run. Some ICs does not pass the test; among all of these some are underclocked, some are thrown away.

2.6.2 Supply Voltage Variation

A power supply is designed to work at a given voltage, but far from ideal, they suffer from fluctuations in the supplied voltage. The nonzero resistance in the supply wires and the self-inductance of a supply line causes voltage drops. For example, when a transistor is switching to high it takes current to charge the gate and if there is a sudden large switching activity it can produce a considerable voltage drop.

The potential difference between the source and the drain is one factor that determines the speed of the current to cross through the transistor. With more potential difference the transistor will charge up the parasitic and load capacitances faster. Hence, with less potential difference, the transistor will do it slower.

2.6.3 Operating Temperature Variation

The electron mobility depends on the temperature. The mobility in silicon decreases when increasing temperature while it is over the critic temperature, below which silicon becomes superconductive and it depends on the doping concentration.

When a chip is operating, due to power dissipation in the MOS-transistors, the temperature can vary throughout the chip. Power consumption – hence, power dissipation – increases with transistor switching activity. This produces what is known as hot spots: regions of the chip where the activity is being very intense and the power dissipation is increasing the temperature of the region over other areas with less activity. This effect implies different delays throughout the chip because hot spots will have larger delays than colder areas due to lower electron mobility.

Temperature not only reduces electron mobility but also increases the threshold voltage of transistors, which also increases the delay. A lower threshold means a higher current and, therefore, less delay.

Mainly, power dissipation is due to MOS-transistors switching activity but, in this technology, there is always present a leakage power consumption and, sometime, short-circuits.

Figure 2.7 shows how delay varies due to PVT variations.

Hence, best case is when we have low process variations, the highest voltage and the lowest temperature. On the other hand, worst case is when we have high process variations, the lowest voltage and the highest temperature.

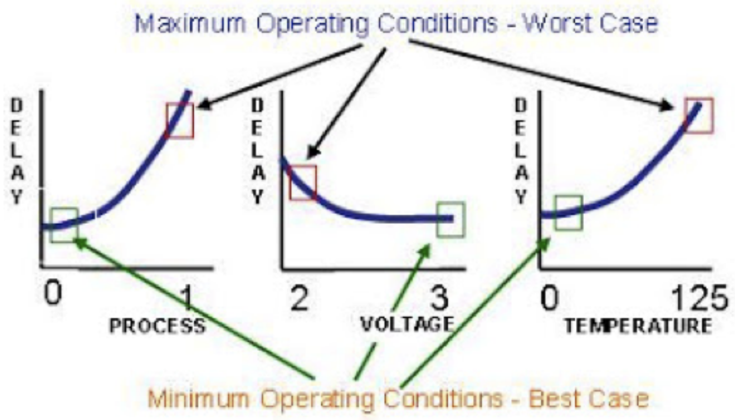


Figure 2.7: PVT over path delay variations.

Chapter 3

Asynchronous circuits

During this chapter will be presented different techniques or methods for implementing asynchronous circuits to further discuss pros and cons of each. But before presenting the asynchronous paradigm it would be useful to explain some concepts about synchronous circuits for a better understanding of the needs of a change in this field.

3.1 Synchronous designs

A synchronous circuit is a digital circuit, which is all synchronized by a global clock signal. The idea is that every change in the sequential elements occurs simultaneous. Having a global clock makes designing circuits much easier. Everything occurs at an exact known time, concurrency can be easily controlled and the performance easily measured. Figure 3.1 shows the block diagram of a common synchronous circuit. It also shows how the sequential logic is governed by a global clock.

Synchronous seemed to be the ideal paradigm in circuit design. So, why do we need to remove the global clock? The answer is Process-Voltage-Temperature variations, which have been explained in section 2.8. With the decreasing size of the transistors, nowadays a few nanometres, the variability in circuits is becoming more and more important as an issue to be taken into account. These performance variations are translated into variable critical paths during execution but in synchronous

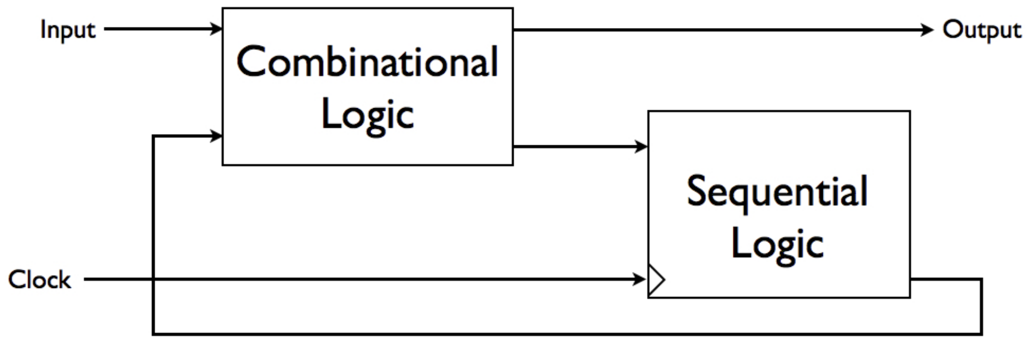


Figure 3.1: Block diagram of a common synchronous circuit.

circuits the clock period remains fixed no matter what. This can became a problem if the critical path exceeds the clock period. If this happens, the stored value in sequential elements would be unpredictable due to timing violations.

Nowadays, this problem is being solved with the addition of conservative margins in the clock period. These margins become a difference on performance up to a generation [9], which makes the circuit run slower than existing technology allows.

To decide the clock period in synchronous circuits the timing analysis is done in the worst-case scenario of the entire circuit' lifetime. For example, if a circuit is intended to work at a 1.2V nominal voltage, the timing analysis has to take into account a case in which the power supply is giving only 1.1V due to power fluctuations and the temperature of the chip is 100 °C. The timing analysis of this scenario will determine the clock period and any scenario worst than this will not be guaranteed to work properly. Furthermore, aging has to be included as another variability cause through time.

3.2 Asynchronous designs

The synchronous paradigm is useful from a specification perspective since it allows the designer to separate timing and functionality. However, implementing an entire design with a global clock has all the problems mentioned above and every day is becoming a bigger problem.

The problem of timing is becoming more and more complex at the time that circuits need more accurate timing analysis. Variability is not the only problem timing has to take into account. The primary source of delay propagation in CMOS

ICs is not the logic but wires – exceeding gate delays by a factor of three or more [11] – and these make difficult to distribute the clock with sharp edges and low *skew* needed for high-frequency operations.

Then it is when appears the asynchronous paradigm. Asynchronous circuits try to solve most of the problems synchronous circuits have.

An asynchronous circuit is a digital circuit, which is not governed by a global clock signal. There are a lot of approaches on how implement asynchronous circuits the best way, each one with its pros and cons. Figure 3.2 shows an example of an asynchronous circuit design. Note that the sequential logic is not governed by a global clock but by some control logic.

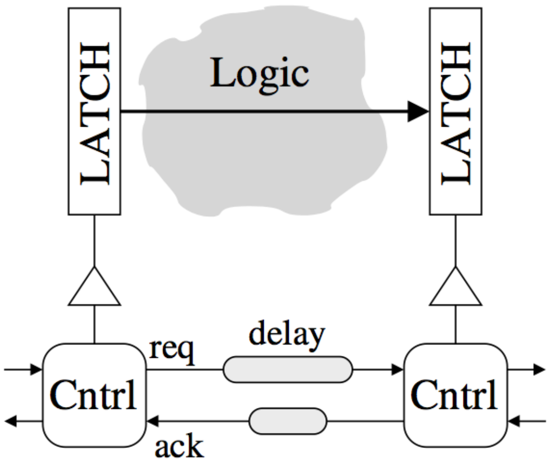


Figure 3.2: Example of an asynchronous circuit design. source: [9].

Sometimes this kind of circuits are called *Delay-Insensitive* because not being ruled by a single signal makes them support different levels of variability without failure. The asynchronous paradigm's main problem is to change too many things to support it. For example, EDA software does not support asynchronous design and without these tools it is almost impossible to design anything significantly big. Furthermore, designers have been decades working with synchronous paradigm and they are used to control their designs with exact timing.

Thereupon, some approaches will be discussed.

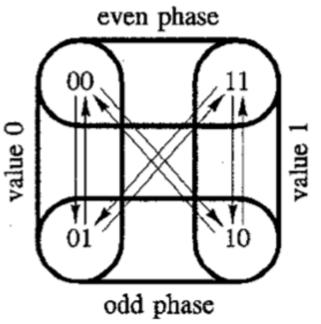
3.2.1 Delay-insensitive circuits: Phased Logic

The first type of asynchronous circuits is the Delay Insensitive, which is the most robust of all the asynchronous circuit models. It makes no assumptions on the delay of wires or gates. In this model all transitions on gates or wires must be acknowledged before transitioning again. This condition stops unseen transitions from occurring. In DI circuits any transition on an input to a gate must be seen on the output of the gate before a subsequent transition on that input is allowed to happen. An example of this model is the following approach, which uses phased logic.

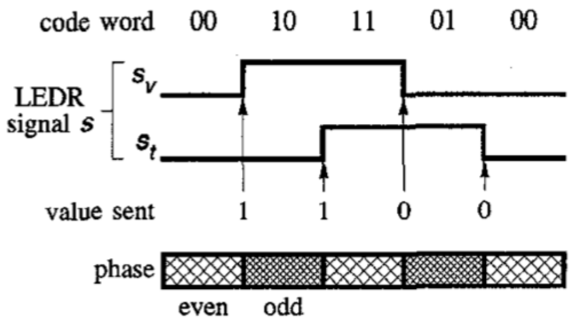
This method [12] uses a two-phased Level-Encoded Dual-Rail (LEDR) scheme. Uses the timing signal and a value signal to encode them into one. This new signal can be even ("00", "11") or odd ("01", "10"); these phases are the source of the name "phased logic".

The method is based on associating a phase with the gate, i.e., a gate can be in the even or odd phase just like a signal. Then, if a gate's phase is even, the gate will recalculate its outputs, or fire, when all of its inputs become even. The term enable is used for a gate that all its inputs match the gate's state. When an enabled gate fires, the gate's phase and the phases of the gate's output signals toggle to the opposite phase.

Figure 3.3a shows how the encoding works and the Figure 3.3b shows how the waveform of the new LEDR signal changes. Phased logic replaces the periodic clock signal with a predictable, rhythmic changing of signal. This provides the same cyclic, deterministic logical behaviour the designer expects from a clocked system.



(a) LEDR code word relationships



(b) LEDR waveform

Figure 3.3

Benefits

- Its support of the synchronous paradigm. The designer can still consider a system governed by a global clock signal and can target a phased logic implementation.
- EDAs tools can be used in the same way except for the synthesis.

Disadvantages

- The synthesis algorithm has to be modified.
- Uses LEDR, which typically leads to significant area overhead – roughly double the area of single-rail designs [6] – and a high power consumption.

3.2.2 Quasi delay-insensitive: Four-phase handshake protocol

Quasi delay-insensitive are a class of asynchronous circuits in which there is no assumption about the delays of any of the circuit's wires or elements but it assumes other properties like some fan-outs – number of gate inputs to which a gate is connected –. An example of quasi delay-insensitive design using the classical handshake protocol as synchronization method is explained as follows.

The implementation of the ARM996HS [4], a clockless 32-bit processor core, uses four-phase handshake and single-rail encode – in order to reduce area and, hence, costs –. Single-rail encoding, as shown in the Figure 3.4a, has the valid/ack signal aside. This makes the design more difficult than the previous approach because there is no implicit handshake, like in double-rail shown in Figure 3.4b. In contrast, the area is reduced considerably by requiring only one wire.

However, ARM designed this processor automatically using a high-level language called Haste. Using a high-level design language the programmer does not have to deal with asynchronous implementation details such as rail encoding or handshake. The Haste program is synthesized into a Verilog netlist based on standard cells. Then, third-party EDA tools can perform their optimizations.

The ARM996HS was tested along with other ARM equivalent synchronous processor, the ARM968E-S. The synchronous processor was clocked at such a speed

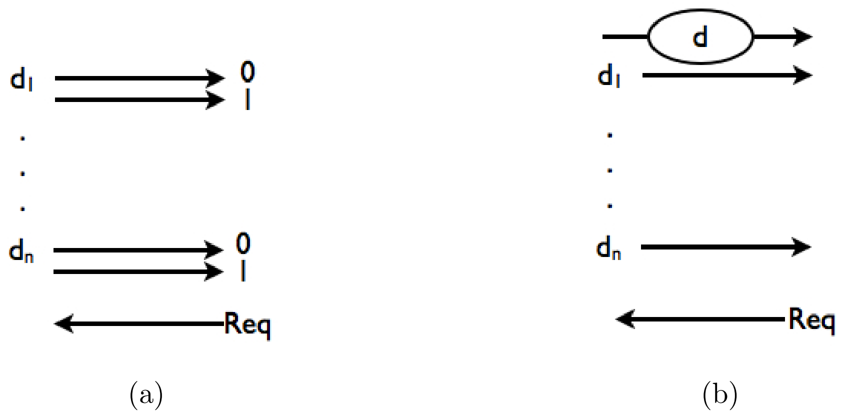
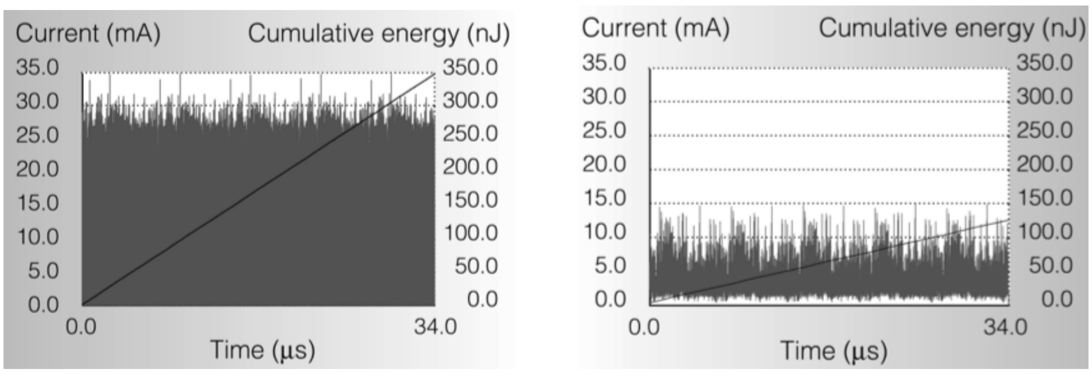


Figure 3.4: Double-rail vs single-rail encoding.

that both processors computed the benchmark in the same amount of time. Under identical benchmark conditions the clockless processor consumes 2.8 times less dynamic power than the clock-gated processor.



(a) ARM068ES (b) ARM996HS

Figure 3.5: Cumulative energy and peak currents for ARM068E-S and ARM996HS.

Under typical conditions (1.2V, 25 °C), the asynchronous design achieves 77 equivalent MHz[1] and under worst conditions (1.08V, 125 °C) it achieves 50 equivalent MHz. The power consumption overt time under typical conditions (and equivalent performance) of both processors can be seen in Figure 3.5. Figures does not represent static power, or leakage, because is similar in both processors. The area achieved is almost the same (89Kgates for the asynchronous and 88Kgates for the synchronous processor).

Benefits

[1]Because of the lack of clock, the term of equivalent MHz is used instead of MHz.

- Proven considerable power reduction. Standby power is zero because it needs no clock to wake up.
- Delay-insensitive.
- No area overhead.
- Transparent to the designer.

Disadvantages

- Haste project has been discontinued.
- It is still difficult to make equivalency with a synchronous circuit and control its performance.

3.2.3 Elastic clocks

Elastic clocks approach [9] presents a method in which an asynchronous design can be performed with synchronous design tools with minor and localized changes in the clock tree generation step.

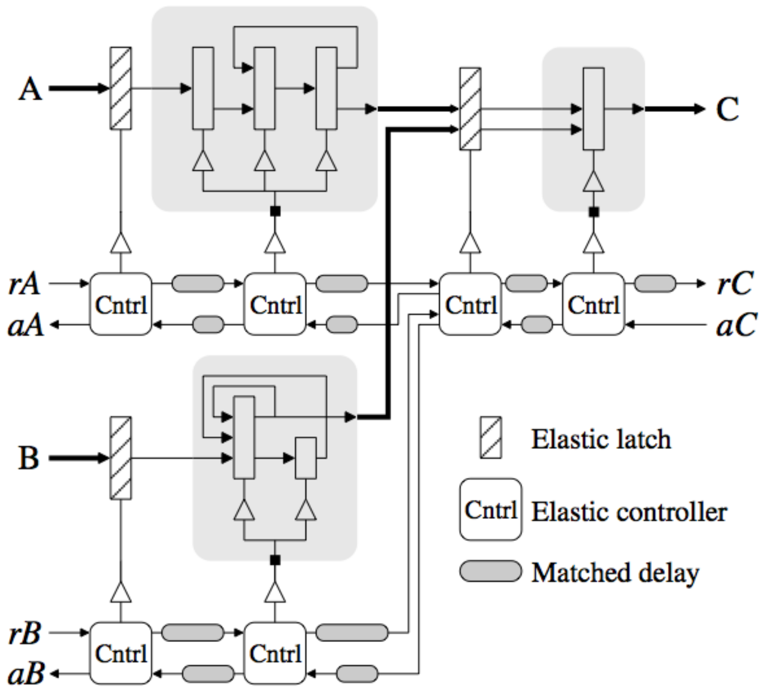


Figure 3.6: Elastic circuit

Elastic circuits are easily generated from an existing synchronous design applying few transformations:

- Circuit is divided into clusters in which there will be a local clock in each.
- Latches are inserted in the boundaries of each cluster in order to synchronize them.
- A local clock and its controller for each cluster substitute the global clock.
- Finally, the circuit is synthesized to meet the timing constraints.

Figure 3.6 shows an example of an elastic circuit that has been adapted from a synchronous design.

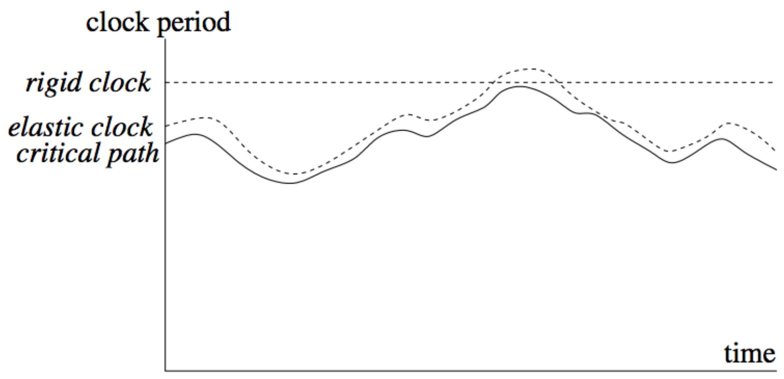


Figure 3.7: Rigid and elastic clock periods. Source: [9]

Figure 3.7 shows the correlation between the critical path and the rigid and elastic clock. As shown, rigid clock does not vary through time while critical path does and this can cause a critical path larger than the rigid clock period, which is totally undesirable. However, the elastic clock, even though it is not delay-insensitive, it varies the same way the critical path does. There is always a margin, but very small compared to the conservative margins added to rigid clock.

Varying the clock the same way the critical path does implies that the elastic circuit can run faster at equal conditions or can run at same speed with worst condition, i.e., reducing the input voltage and, hence, the power consumption.

Benefits

- The design flow remains intact.

- Even though being an asynchronous circuit – and having its benefits – the design is conceptually synchronous.

Disadvantages

- Elastic circuits still need a safety margin added to the critical path to determine the clock period.

Chapter 4

Proof of concept: Closed-feedback loop

The project has two distinct parts and different tools and libraries have been used in each one. One is the design of the circuit for the FPGA, including the ring oscillator and the test circuits. The other one is the Linux application, which shows information about what is happening inside the FPGA. The software shows performance or workload charts among others and can control some parameters like the reference clock.

In this chapter is presented our implementation of a closed-feedback loop using elastic clocks method for implementing an asynchronous circuit.

At Figure 4.1 can be seen the aspect of the final platform. It includes the closed-feedback loop and the computer, which controls some parameters and shows in a graphical mode the information obtained from the closed-feedback loop.

During this chapter the FPGA used as development platform is presented along with a brief introduction to the power supply used. Also, it is presented and explained the techniques used, like the elastic clock and adaptive voltage scaling. Later, some the details of the hardware implementation are given in order to understand how everything works. Finally, the software application is presented and briefly explained.

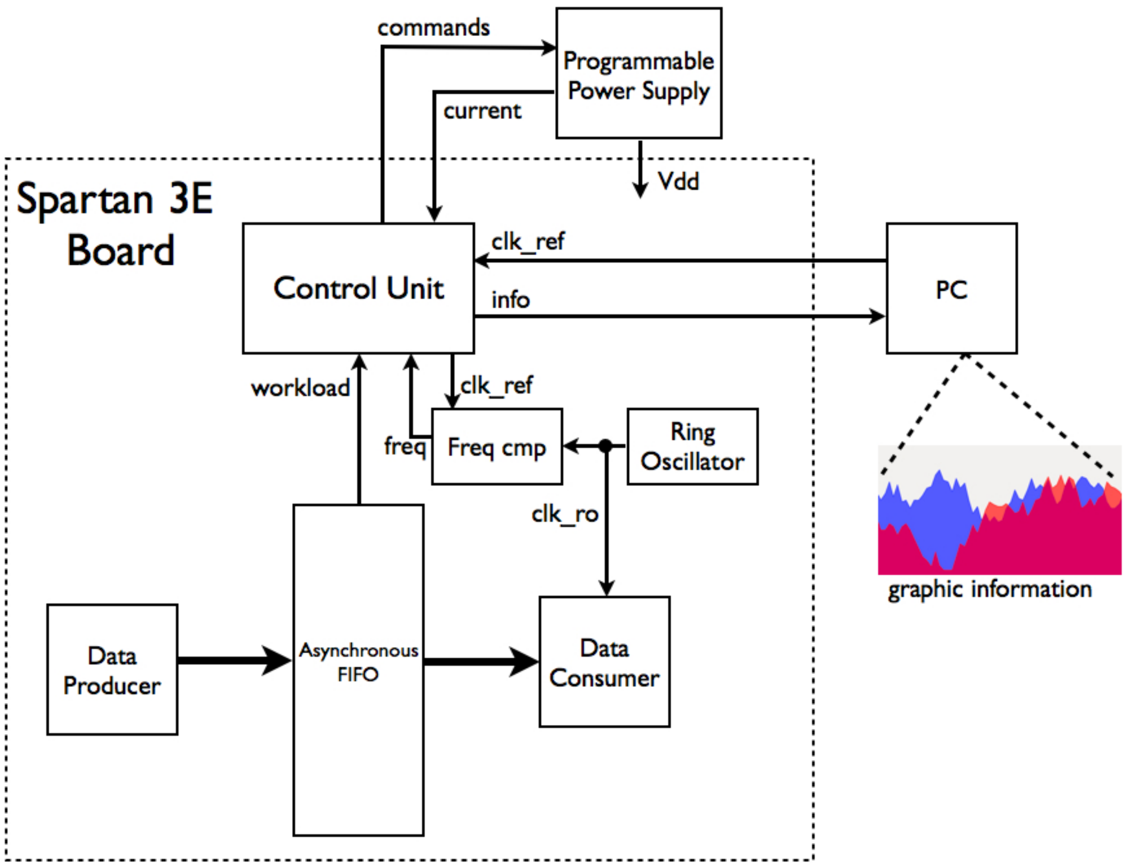


Figure 4.1: Block diagram of the closed-feedback loop system developed.

4.1 Xilinx FPGA

FPGA stands for Field-Programmable Gate Array and it is an integrated circuit designed to be reconfigured after manufacturing. FGPAs contain programmable logic components called *logic blocks* and a hierarchy of reconfigurable interconnects that allow the blocks to be wired together. Logic blocks can be configured to perform complex logic functions in form of a truth table inside a *Lookup Table* (LUT) in the logic block, which can contain also simple logic gates – most common are AND or XOR – and some contain memory elements, such as flip-flops.

The project has been developed and tested using Xilinx FPGAs. Actually, two different models have been used: Xilinx Spartan-3E and Xilinx Virtex-5 (XUPV5); each one with different specifications. However, the Spartan-3E has been used during the whole project and it is the test and develop device, the XUPV5 has been restricted to test more complex and bigger designs.

The Spartan-3E board, manufactured by Digilent Inc., has 500k gates. From all the components it has, the most important and the used for the development of the project are: two RS-232 connectors – DCE and DTE – and an Analog-to-digital converter (ADC). Also has a JTAG USB interface, which allows debugging and programming the FPGA through USB.

4.2 Asynchronous design

Among all the approaches discussed in the Chapter 3 about asynchronous circuits we have decided to implement the elastic clocks. The main reason of choosing this methods is the possibility of working as if it was a synchronous design.

In this method, a *ring oscillator* substitutes the global clock in order to reduce the safety margins added to the period over the critical path and thus reduce power consumption. Also, it is used to perform a technique for reducing energy consumption called *Adaptive voltage scaling*. Using this technique, the input voltage to the circuit is managed – increased or decreased – taking into account the workload in order to be as efficient as possible.

4.2.1 Ring Oscillator

A ring oscillator is a circuit composed of an odd number of NOT gates whose output oscillates between low and high levels. The feedback to the last output to the input causes oscillation.

Each gate of the oscillator has a certain delay under certain conditions. This implies that the period of the oscillator is at least twice the sum of the individual delays of the gates – delay of the wires has to be taken into account –. Also, if conditions like temperature or voltage change – variability – with the corresponding change of delay of the logic, then the gates of the oscillator will experiment the same or similar delay variation than the rest of the logic. The variation is not always the same because temperature or voltage fluctuations are not uniform throughout the whole circuit.

Thanks to the ring oscillator's adaptability to variability it is possible to reduce the conservative margins added to guarantee the reliability and to ensure that the clock period will be always larger than the critical path. These margins can be highly reduced but not completely removed due to slightly different conditions throughout the same chip. The reduction of the margins leads to a reduction of the power consumption.

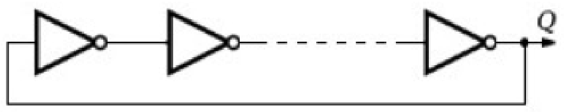


Figure 4.2: Ring oscillator implemented with AND gates.

Figure 4.2 shows the aspect of the ring oscillator's implementation.

4.2.2 Clock domaing crossing

Even though a ring oscillator is intended to govern the vast part of the circuit, some applications cannot be used with an elastic clock. There are functions or protocols that need an exact timing. For instance, the RS-232 protocol implemented in this project needs a fixed clock to generate the exact baud rate that will determine the bit time and if it varies considerably the reception could fail.

Each clock of a circuit determines what is called a clock-domain because that single clock governs all the logic within the boundaries of its domain. When data needs to cross a domain it cannot be done straightforward because glitches and metastability can appear and corrupt the data. Clock-domain cross elements are needed by the data to cross error free.
In our case, the cross element has been an asynchronous FIFO with a clock to read and a different clock to write in it.

Glitch A glitch is an undesired transition that occurs before the signal settles to its intended value. It is dangerous when occurs between sequential elements because these are triggered by a transition and a glitch can trigger a flip-flop at an undesired moment and it would store wrong data.

Metastability Metastability describes the extended duration of a change of state. In electronics it is a problem because the time needed for a signal to switch from one state to another is uncertain and can be prolonged indefinitely. If the circuit is vulnerable to metastability, it can get stuck in an undesirable state.

4.3 Adaptive voltage scaling

The main purpose of the project is to reduce the power consumption with asynchronous circuits. Using a ring oscillator as clock, there are two main ways to do it. The first one is reducing the margins added to the clock period, which gives a constant saving, in terms of percentage. The other way is varying the input voltage. A reduction of the input voltage would lead to a reduction of the power consumption but also an increase of the gate delay – hence the clock period –, which is easily and robustly managed by the ring oscillator, guaranteeing reliability.

This is a common technique in synchronous circuits for reducing power consumption. However, it has to be managed in a slightly different way. With a rigid clock, an input power reduction needs to be preceded by a reduction of the frequency and an increase in frequency needs to be preceded by an increase of the input voltage, otherwise it could lead to a timing violation due to an increment of the critical path, which could overcome the clock period. This process has to be strictly managed and it is not instant because only changing the frequency on a PLL – *Phase-Locked Loop* – is not a trivial operation and it may take a significant amount of time. Elastic clocks take even more benefits from this technique.

The Spartan-3E does not have a variable power supply, which means that an external one is needed. To be able to use an external power supply, the board has to have a way to cut the internal supply and pin the new one. Looking at the schematic view of the board, Figure 4.3 and found in the datasheet, we see how the board has different input voltages for different internal devices. The interesting one is the one at 1.2V and luckily for us, there is a jumper between the 1.2V input and the board. This means that if we remove the jumper J7, opening the connection, we would be able to pin the voltage from an external power supply.

4.3.1 Agilent E3649A

The jumper give us the possibility to connect an external power supply but we need a variable power supply. The chosen power supply is the Agilent E3649A, manufactured by Agilent, is a power supply with the capability of modify the output voltage through a RS-232 interface, which has been use to control the voltage.

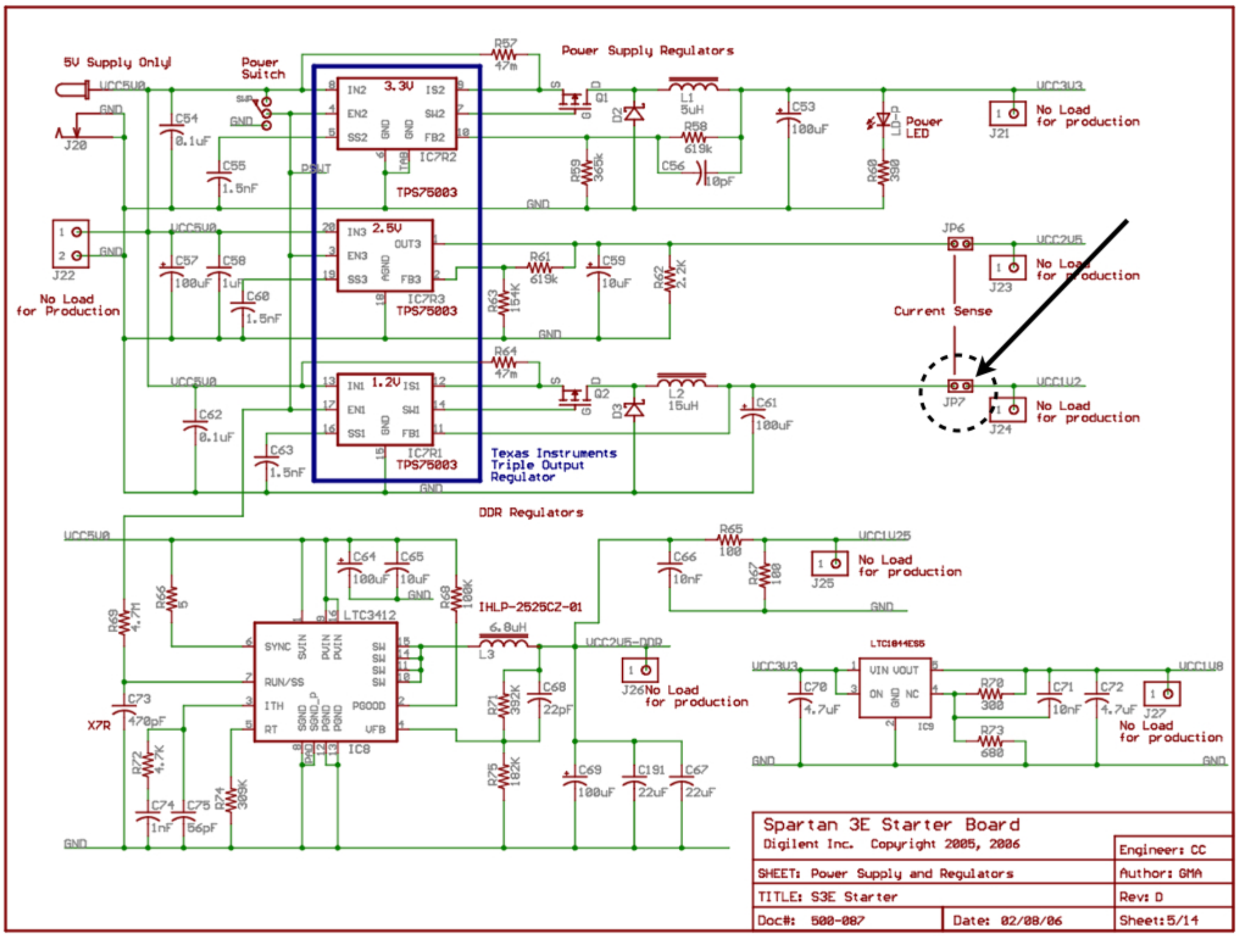


Figure 4.3: Schematics view of the FPGA Spartan 3E Board.

The power supply can provide from 0 to +36.05V. The variations of the voltage are done in steps of 10mV minimum and it takes less than 10ms to complete the request. However, according to the datasheet, the time between requests has to be greater than 180ms, which corresponds to a lot of clock cycles. Thus, this constraint cannot be ignored during the design.

The control of the power supply is done easily sending through the RS-232 interface simple commands. The syntax is very easy; it consists in a string in uppercase and always ended with a new line character. Some of the most important are: “STEP: 0.1” – to define the step length –, “STEP UP” – to increase voltage one step – or “STEP DOWN” – to decrease voltage one step –.

4.4 Closed-feedback loop

Once we have an FPGA with the possibility to connect an external power supply and a programmable power supply we have everything we need in order to implement and test the *closed-feedback loop*. Closed-feedback loop is the cornerstone of the project. It consists in any circuit governed by an elastic clock, implemented with a ring oscillator, and a frequency comparator in order to control the input voltage taking into account some parameters. This is technique is called Adaptive Voltage Scaling using a closed-feedback loop. There are to parameters that can be used by the control unit to determine the way power will be scaled:

- Workload: the circuit calculates the amount of work it has to do and powers up or down depending on it. When the workload exceeds a given threshold, the power input is increased. On the other hand, if the workload falls below another threshold means that the circuit is going too fast, hence, consuming more than necessary. If this occurs, analogously to the other case, the power input is decreased in order to make the oscillator run slower.
- Reference clock: A reference clock is defined and its frequency is used to compare it with the oscillator's frequency. If the comparator finds out the oscillator is going slower than the reference clock, the input voltage is increased. On the other hand, if the oscillator is going faster, the input voltage is decreased.

Note that the adjustment of the input power is not suitable for resolving metastability problems because of the huge latency of the power supply. The adjustment is only for power consumption and performance purposes.

The circuit is divided in two parts:

- The circuit that does real work and produces the control data for the other circuit. It is called working subsystem.
- The circuit that monitors the first one – sending information to the PC – and also adapts the power supply at its optimum state. It is called monitoring subsystem.

Figure 4.4 shows the circuit that uses the ring oscillator as a clock. In this design, there is a circuit that produces data, which in this case is governed by the fixed clock

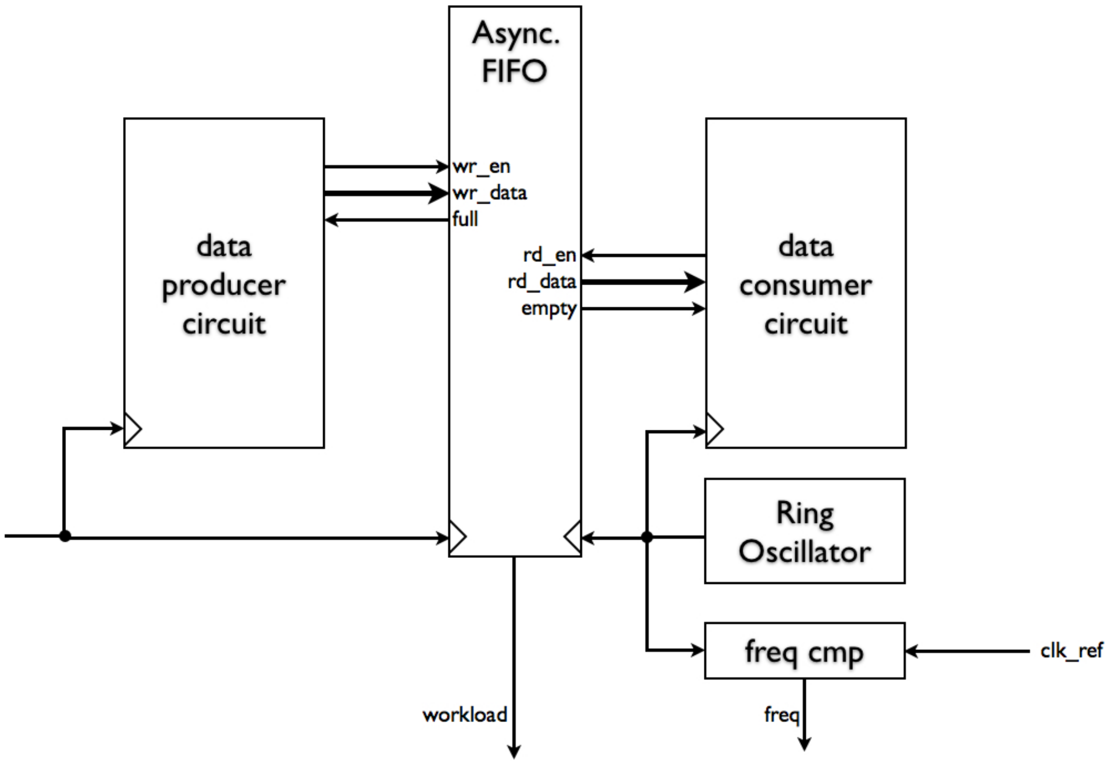


Figure 4.4: Conceptual block diagram of the working subsystem.

of the FPGA and another circuit, governed by the ring oscillator, consumes the data and operates with it.

This is the general conceptual design. It can vary depending on the characteristics of the circuit. The two subsystems – producer and consumer – can be balanced in any way. For example, in the design used for testing the approach – discussed later – the producer only reads the memory and the consumer does all the work. On the other hand, in the real case proposed with the IIR, an ADC produces the data and stores it in the FIFO and the IIR consumes it.

There is an asynchronous FIFO between the producer and the consumer used as a cross-domain element. On one side, there is the logic for writing a value into it and is governed by the fixed clock – clk signal – and on the other side, there is the logic for reading the stored values. The ring oscillator – clk_osc signal – governs the reading logic. Therefore, the FIFO has two clocks. A new value is stored with one-cycle latency and a new value is also read with one-cycle latency – measured with their corresponding clocks –. The width is variable depending on the application but the depth has been fixed to 256 elements. One of the most important information that outputs the FIFO is the number of elements inside it, which is the actual workload.

The ring oscillator works as previously said but it has a new element to monitor its performance. In order to know the exact frequency of the oscillator at any time the following method is followed:

- A number fixed clock ticks is defined, from now max_ticks value.
- Two counters are added to the design. One increments its value with each FPGA clock and the other with each ring oscillator clock.
- When the max_ticks is reached, the value in the ring oscillator counter is saved.

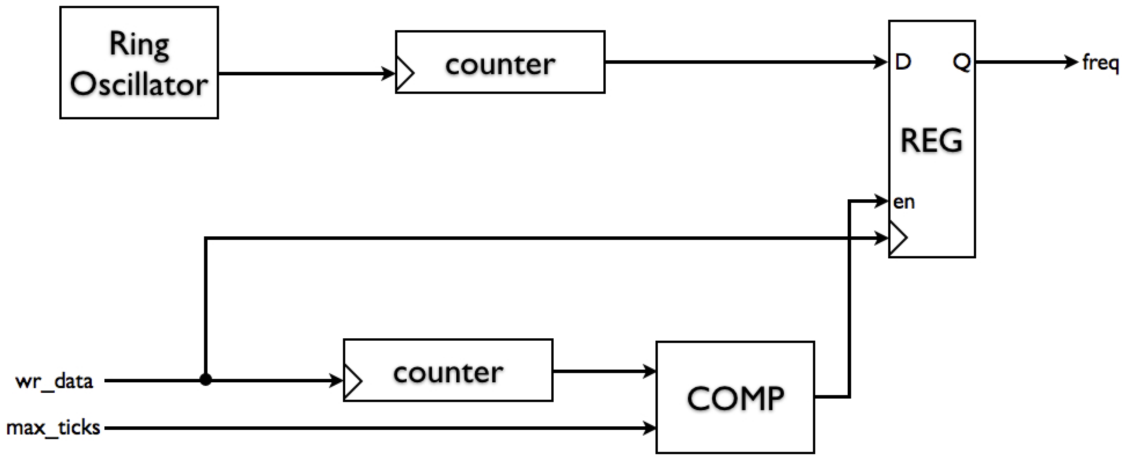


Figure 4.5: Conceptual block diagram of the frequency comparator circuit.

This value obtained from the circuit does not represent the frequency of the ring oscillator but the relation between the two clocks and gives enough information to calculate the exact frequency. Since the number clock ticks is known, the next equations can be used to determine the exact frequency:

$$T_{osc} = \frac{max\_ticks * T_{clk}}{osc\_ticks}$$

$$f_{osc} = \frac{1}{T_{osc}}$$

However, in order to avoid complex logic to work with floating point and to avoid dividers, the FPGA works with the number of ticks of the oscillator, which is an integer. Thereby, it has to be defined an upper and lower threshold from max_ticks and they will determine if the oscillator is going too fast, too slow or at its optimum speed.

Finally, the workload and the number of ticks of the oscillator are driven to the next part of the closed-feedback: the monitoring subsystem. The monitoring subsystem is responsible for using the information obtained by the work subsystem.

The subsystem has mainly three elements: the control unit and two RS-232 interfaces to send and receive information from the computer – DCE connector – and the power supply – DTE connector –. The RS-232 works at 9600 bauds for transmitting and receiving data from the computer and from the power supply. For further details of the RS-232 implementation refer to the Appendix B.

There are three types of information: the workload, the frequency and the information about the power consumption. All of them are sent to the computer but they are also used to make decisions. Figure 4.6 shows the conceptual aspect of the monitoring subsystem along with all its blocks.

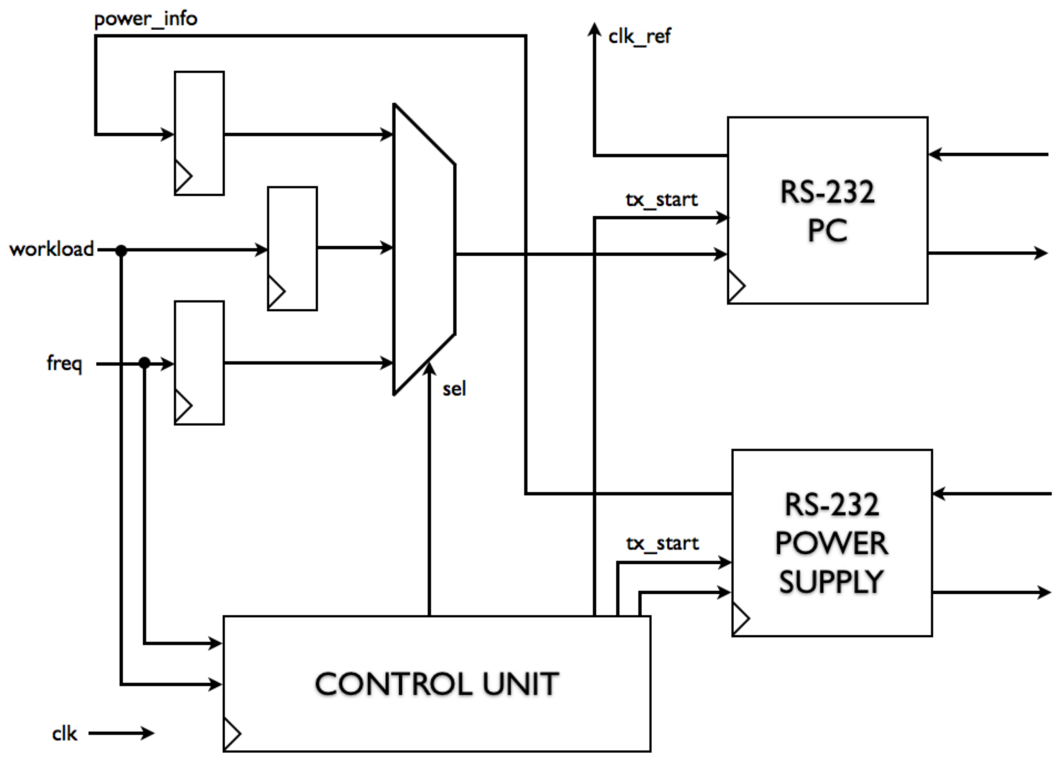


Figure 4.6: Conceptual block diagram of the monitoring subsystem.

There are two operating modes: workload or reference frequency:

- Workload: the power supply is controlled taking into account the workload of the work subsystem.
- Frequency: the power supply is controlled taking into account the reference frequency, which is obtained from the computer through the serial port.

Depending on which mode is the circuit in a given moment, the control unit will take into account one or another. The information retrieved from the power supply is used from the control unit mainly to know if the maximum or minimum voltage levels have been reached, i.e. if it needs to decrease the input power but the minimum voltage level have been reached, the control unit cannot allow it and vice versa.

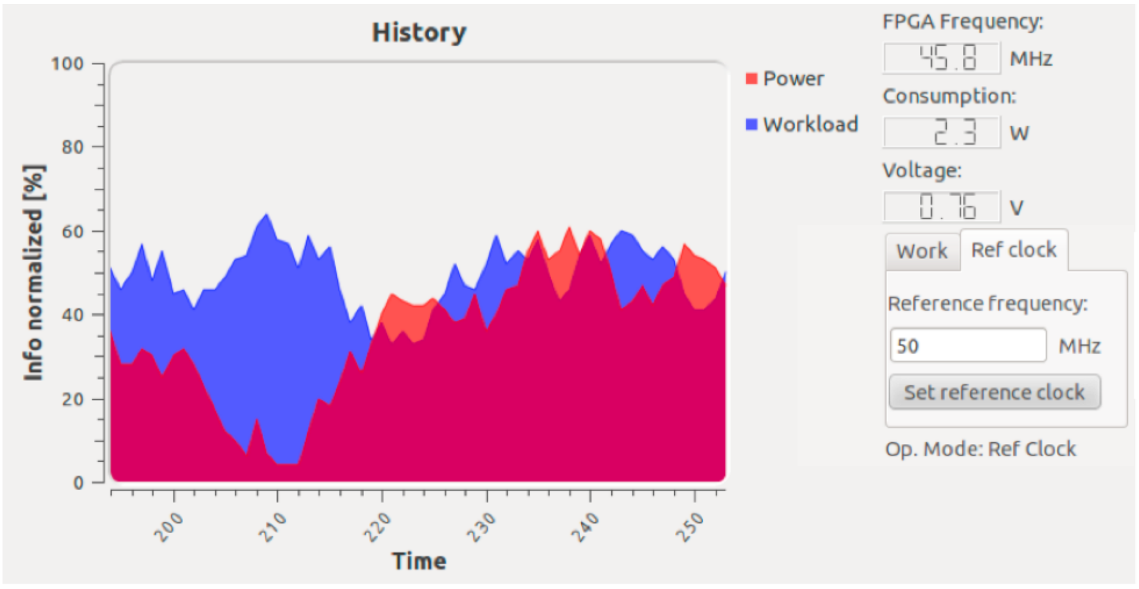
The control unit controls the multiplexer with all the information to be send to the computer with the select signal that changes every time a transmission to the computer finishes. From the computer, the only data retrieved is the reference frequency, which is directly driven to the frequency comparator. In a similar way, the port connected to the power supply is controlled.

As already stated above, the power supply has timing constraints because it cannot accept requests in gaps smaller than 180ms. The control unit has to guarantee that this constraint will not be violated. Thus, it has a counter used as a time, which is reset every time a transmission to the power supply finishes. When the control unit decides to vary the input power, it sends a new command if enough time has passed.

4.5 Linux applicationn

The FPGA design has been implemented using the elastic clocks approach. However, there is no way to know how is it working, if it is working better or worse than the original implementation using a fixed clock. For this reason, it was needed something else to help to what is happening inside the board and, also, adjust some parameters.

Figure 4.7 shows the actual aspect of the application. It has been implemented in C++ using Qt, a cross-platform application framework. However, the application does not run in windows directly because it uses system calls to communicate



FPGA Frequency: 45.8 MHz
Consumption: 2.3 W
Voltage: 0.76 V
Reference frequency: 50 MHz
Op. Mode: Ref Clock

Figure 4.7: Graphical interface of the FPGA Monitor software.

through the serial port. These calls are O.S. dependent and they may have to be translated in order to use the application in environments others than GNU/Linux.

The interface is divided in three distinct parts: plots, information panel and selection of the operating mode.

4.5.1 Plots

The plots part is a widget inside the interface. It has been implemented using a library called Qwt [3], which is intended to help to develop Qt widgets for technical purposes, like 2D plot widgets. The widget helps to view more easily the information retrieved from t he FPGA. It contains two plots: the red one shows the power consumption and the blue one shows the workload. Clicking on their names in the plot legend each one can be enabled/disabled individually.

The information is updated every second with the last data obtained from the FPGA. The X-axis represents the time (seconds) while the Y-axis is the value normalized in %. It can be shown easily in percentage because the maximum and minimum of power and workload are known value. For example, the minimum workload is 0 while the maximum is 255. The maximum and minimum power can vary depending on the application, after doing some timing analysis and testing them, the margins are established. For instance, it could be 0.70V as the minimum and 1.2V as the maximum. Notice that the monitoring subsystem of the circuit uses a fixed clock. That subsystem should be small and should not increase margins considerable but it has to be taken into account during establishing the lower threshold of the downscaling voltage.

4.5.2 Information panel

The information panel is used to show all the information obtained from the FPGA or the information calculated from the data obtained. The aspect of the information panel is shown in the Figure 4.8.

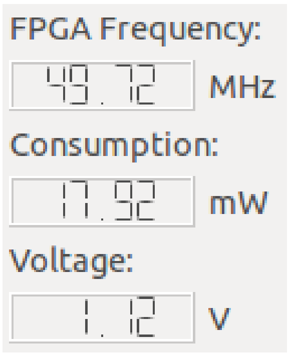


Figure 4.8: Information panel.

The information known about the FPGA are basically the frequency, the workload, the voltage and the current. Since the workload is better understood with a plot, there is no need to show it again in a numerical form.

The frequency is not obtained directly because it would imply working with real numbers for the FPGA and it is desirable to not to do so, since it would need complex logic and it would lead to a larger overhead. Besides, the computer can do the operation extremely easily once it has the number of ticks of the oscillator because the number ticks of the fixed clock and the frequency of the clock are known values.

The computer knows the initial voltage of the FPGA and from here the FPGA sends a relative number of steps done. For example, if the minimum voltage is 0.70V, the maximum 1.2V and the steps are 0.01V there are 50 steps from the minimum to the maximum. If the initial voltage is 1V, the initial relative number would be 30 and then the FPGA increases and decreases that number each time a step up or a step down has been performed. This method avoids asking to the power supply each time what is the actual voltage, which is slow and expensive.

Once, the voltage is known, the FPGA should only ask for the current. The equation of the dynamic power dissipation of a chip is the next:

$$P_{dynamic} = a * f_{clk} * V_{dd}^2$$

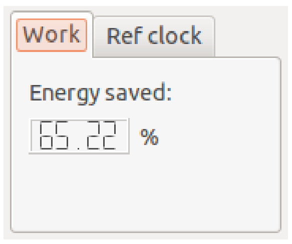
Being a the switching activity, f$_{clk}$ the frequency of the clock, CL, the capacitance of the circuit and VDD the supply voltage. However, most of this parameters are very difficult to know in a FPGA, this is why the more simple P = V * I equation has been used. Then, a simple operation gives us the power consumption of the FPGA and it is shown in the information panel.

4.5.3 Operating modes

The application allows putting the FPGA in two different operating modes:

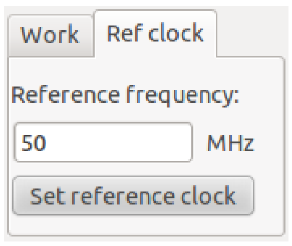
- Work mode: In this mode the FPGA tries to achieve an optimum power consumption taking into account the workload at each moment. During this mode, it is shown the energy saved in percentage respect to the solution with the fixed clock.

- Reference clock mode: A reference frequency is introduced, it is sent to the FPGA and it tries to equal that frequency varying the voltage. Similar to the frequency received from the FPGA, the reference frequency cannot be sent directly and it has to pass through the inverse operation to obtain the number of ticks that would represent the reference frequency.



Energy saved: 65.22 %

(a) Workload mode



Reference frequency: 50 MHz

(b) Reference mode

Figure 4.9

The selection of the operating mode is done clicking on the corresponding tab. The Figure 4.9a shows the view of the work mode and the Figure 4.9b, the view of the reference clock mode.

Chapter 5

Testing the asynchronous approach

In this section our asynchronous approach is tested in order to know the improvements achieved over the synchronous homologue.

The project consists in showing the benefits of asynchronous circuits over the synchronous circuits. Also, the project is intended to implement a proof of concept of a closed-loop feedback system trying to reduce energy consumption with voltage scaling and asynchronous circuits along with a monitoring platform that has been developed. However, before testing the whole platform we need to proof the benefits of the elastic clocks and to show why they have been chosen.

For this reason, during this test have been used a test circuit without all the subsystems for monitoring in order to reduce overhead – power and area overhead –. Even though a small subsystem for knowing the working frequency is still in the design.

5.1 Test circuit: 16-bit RISC

The project needed a simple and versatile test circuit. During the course of the project, some test circuits were developed and used. However, some were way too complicated or too rigid. We needed a circuit easy to analyse and even better if it could be versatile, to test it in different ways. All these things and the interest of the author for the architecture drove to the development of a tiny microprocessor, which can be programmed via machine code (ASM).

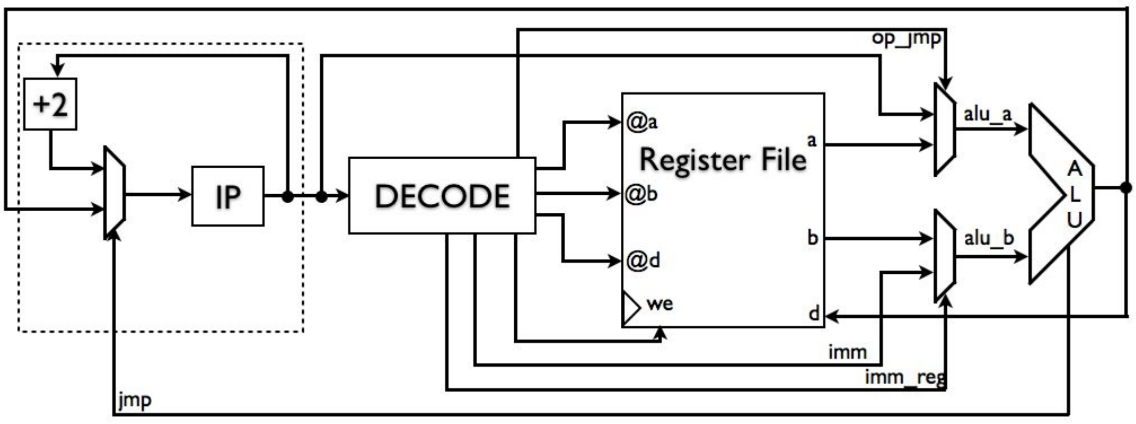


Figure 5.1: Pipeline of the 16-bit processor used for testing.

The microprocessor is a 16-bits RISC (Reduced Instruction Set Computing). The ISA (Instruction Set Architecture) has 16 instructions, including arithmetic, comparisons and conditional and unconditional jumps.

The pipeline is not segmented and is the standard pipeline in RISC machines without superscalar nor multi-cycle operations: Fetch, Decode, Register File, ALU and again Register File in order to write back the results.

The tests has been performed using a simple piece of code, which is basically an infinite loop:

```
MOV   10, R0;        //1111000010100000
MOV   1, R1;         //1111000010100000
ADD   R1, R1, R1;    //0000000100010001
SUB   R0,  1, R0;    //0101000000010000
BNZ   -8;            //1110111111000000
JMP   @0;            //1100000000001111
```

Table 5.1: Machine code loaded in the RISC processor used for the tests.

Table 5.1 shows the piece of code used during the tests. Since no compiler was developed the software had to be coded directly using machine language at its lowest level, which is the binary code. The table also shows code in its assembler equivalent form. Basically it initializes R0 as index of the loop – initialized to ten because the loop will be performed backwards – and R1 is used as a counter. Actually, the loop only counts in R1, adding one each time. Then, the index is decreased and BZ is a conditional branch and branches if the result of the previous operation was different to zero. If it was zero, i.e. the loop finished, the JMP instruction, an unconditional jump, is responsible to jump to the start of the memory and, hence, start again the loop.

The error detection is done checking each cycle the register file and the result outputted from the microprocessor to higher-level hierarchy. Since the microprocessor performs an infinite loop, results and processor state in each cycle are previously known. Once the circuit has been presented, tests will be presented.

5.2 Tests

Tests try to cover all the aspects that can be measured within an FPGA and that are significant in circuits design.

This section has been divided according to the different tests performed. The circuit has been tested introducing noise to the input voltage to measure its robustness and using power scaling to prove what will be used for the closed-loop feedback in order to reduce energy consumption.

The results are finally compared to the results with the same tests but with the synchronous implementation. As said before, the frequency monitoring subsystem will be left in the testing design in order to know at which speed is the elastic clock working, which will give a better perspective of the results obtained.

5.2.1 Overheads

The asynchronous design has a small overhead in terms of area because it has to add the oscillator, which is a variable number of NAND gates, 64 in this case, but since it is who produces the clock no timing overhead is added with it.

For these tests has been added the frequency monitoring subsystem and, with it, the RS-232 DCE subsystem in order to send this information to the computer and catch up the data. These elements add overhead in terms of area and, more important, in terms of critical path because they include synchronous elements, which are very susceptible to variability. However, it has been considered that with such a small inclusion the overhead would be relatively small over the benefits of knowing the clock frequency at any moment.

5.2.2 Timing analysis

Before starting the tests, a timing analysis is needed in order to know how are the two designs to be compared. The analysis of the synchronous design is very important since it is the design that is supposed to fail first and we need to know when is that supposed to happen. Unfortunately for us, the timing analysis that allows the available tools, Project ISE from Xilinx in this case, are only performed in the voltage range of 1.04V – 1.3V, which is insufficient for what we need.

The timing analysis has been done taking 1.2V as the input voltage and 50 °C as the temperature of the circuit. With these parameters the analysis predicts 9 nanoseconds of critical path, but this is always a very pessimistic value.

5.2.3 Power scaling

Power scaling is the base of what will be the project. The closed-loop feedback presented uses the power scaling as its cornerstone for reducing the energy consumption. Within this test, the test circuit is subjected to the power scaling technique in its asynchronous form and its synchronous form. The results should show how the synchronous form does not support the same reduction of its input voltage and it fails sooner. Actually, if the initial oscillator period is enough for the critical path it should not fail at any case only with power scaling. However, it still needs margins if noise is introduced – not in this test –.

First of all it would be interesting to know the correlation of the input voltage the frequency of the circuit. Figure 5.2 shows the frequency of the elastic clock and the fixed clock while scaling the input voltage.

As we can see in the chart of Figure 5.2, the fixed clock remains at the same frequency no matter how much voltage gets at the input. On the other hand, the elastic clock frequency – fixed at approximately 50MHz at 1.2V – has a lineal correlation with the input voltage. Since the ring oscillator is composed by the same circuitry than logic circuit, we can deduce the same correlation but inverted for the critical path, i.e., less input voltage implies a larger critical path and they

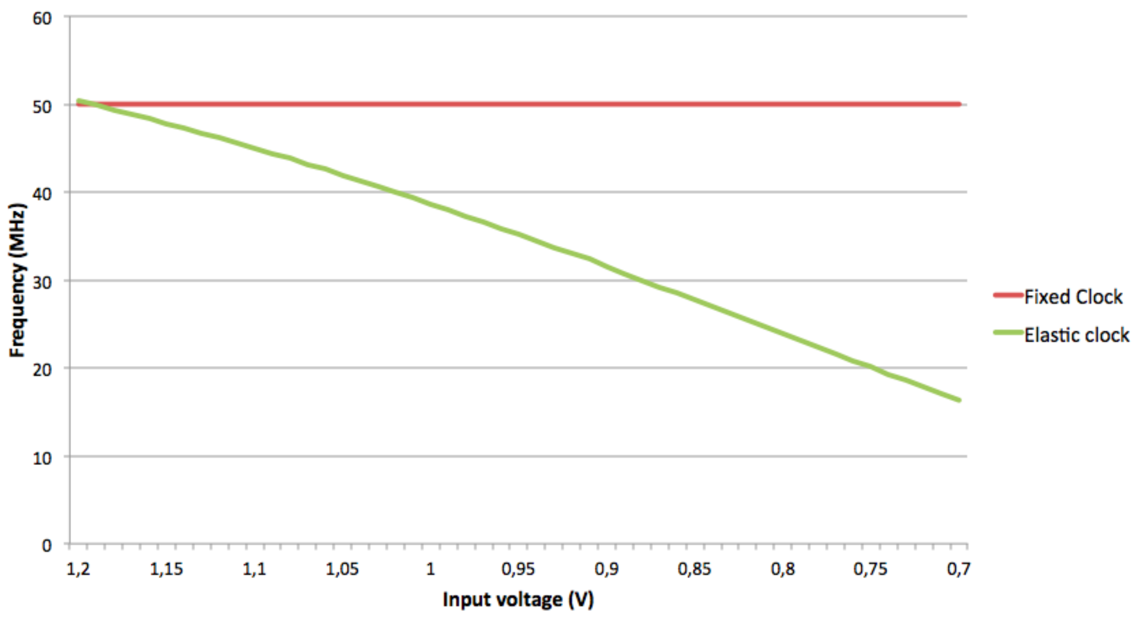


Figure 5.2: Frequency / Voltage correlation chart.

are inversely proportional in a lineal way. After 0.69V frequency plummets because this is the point the circuitry used for sending the information through RS-232 fails – it uses the fixed clock –.

We also know at which point the synchronous circuit fails and the period of the elastic clock at every moment. With all this information, and the critical path at 1.2V, the next chart has been done.

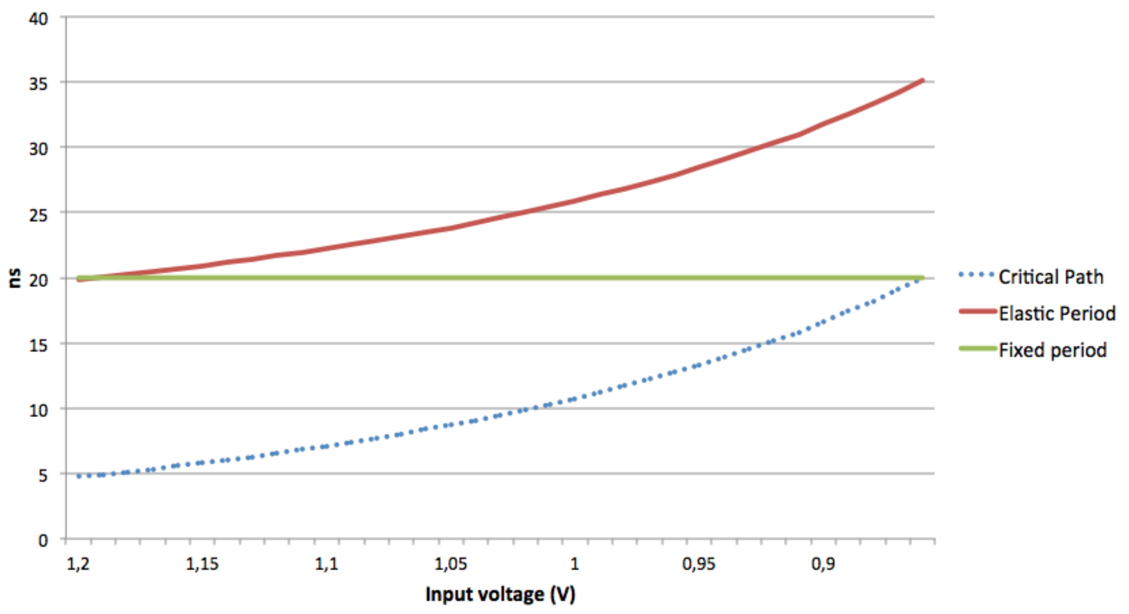


Figure 5.3: Evolution of logic delay over voltage variation chart.

In the chart of the Figure 5.3 we can see the period of the elastic and the fixed clock and the critical path, which is a total estimation, not the real value because the only known value is where the critical path overcomes the fixed clock period, i.e., critical path is larger than 20 nanoseconds and from here, it has been taken the same behaviour for the logic of the circuit and the logic of the ring oscillator, which composes the elastic clock. Taking this into account, we can see how the period of the elastic clock grows the same way the critical path of the circuit does. This implies that no error can occur if there is no noise because the period and the critical path lines will never cross. However, we can see how the critical path overcomes the period of the 50MHz fixed clock and that is the point where the circuit fails.

The difference between the period of the elastic clock and the critical path is quite large and it could have been shrunk choosing a lower period at 1.2V. It was chosen in order to compare the two clocks at the same starting point. However, choosing a lower period, always with a safety margin, would produce similar results: no failures and the same form of the period line.

Once we have this information we can proceed to test the circuit. The method followed has been simply downscaling the input voltage – manually, because there is no circuitry for it in order to reduce overhead – from the same starting point – 1.2V at 50MHz –. The downscale step is 0.01V and at each step power consumption – in the form of current and voltage – has been registered.

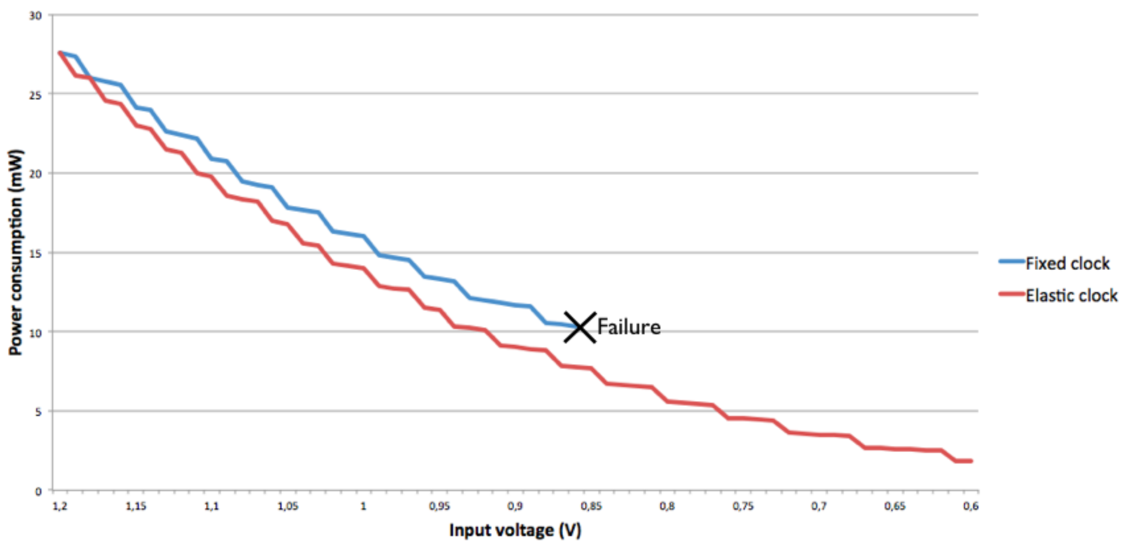


Figure 5.4: Power consumption over voltage variation chart.

Figure 5.4 shows the power consumption of the asynchronous circuit and its synchronous counterpart. Below 0,86V the synchronous circuit fails but the asynchronous continues working. Once the synchronous fails and the asynchronous still accepts downscaling is obvious that the asynchronous will achieve lower consumption but why is it lower at the same voltage? It is because of the frequency. CMOS technology only dissipates and consumes power while switching – in addition to leakage power – and all the sequential logic switches at every cycle, which means that lower frequency, lower switches and, hence, lower power consumption. As seen before, downscaling voltage leads to a lower frequency of the elastic clock and this is what makes the asynchronous to consume less.

5.2.4 Noise

Noise is an important factor that makes the performance of a circuit changes when suffers from it. Noise can be suffered in form of EMI (*Electromagnetic interference*) or in form of voltage variations because of drop voltages throughout the circuit or because of a supply voltage that becomes overcome by sudden necessities of the circuit. The tests are performed using a signal generator that can produce signals of different forms within the range of 20Hz – 20MHz. The signal of the signal generator is introduced in the supply circuit of the board the same way through the jumper J7 – the same used for using the external supply voltage –.

The initial voltage is set to the minimum at which the synchronous design still works without failure. For this test the critical path has been artificially enlarged to make it fail sooner. Under this conditions, the initial input voltage selected has been 1,13V. Initially, the noise is 0 and from here, the noise is being incremented until the circuit fails and the noise level registered. Then, the voltage is increased with the accuracy the supply voltage allows and the noise is again incremented until the another failure is reached. This has been done successively roughly forty times, reaching a voltage level of 1,55V and a noise of 1,170V peak to peak.

The induced noise is being generated with the signal generator Agilent 33220A. The waveform is a sinus and oscillates at a fixed frequency. In the next cases the frequency chosen has been 1KHz and during the test the amplitude (voltage peak to peak) is being changed, which is equivalent to introducing noise.

The test has been performed using the synchronous circuit, running at 50MHz, and using the elastic clock running also at 50MHz when working at 1,13V. As the noise is periodic, it makes the mean frequency to remain at 50MHz.

Figure 5.5 the results of the test:

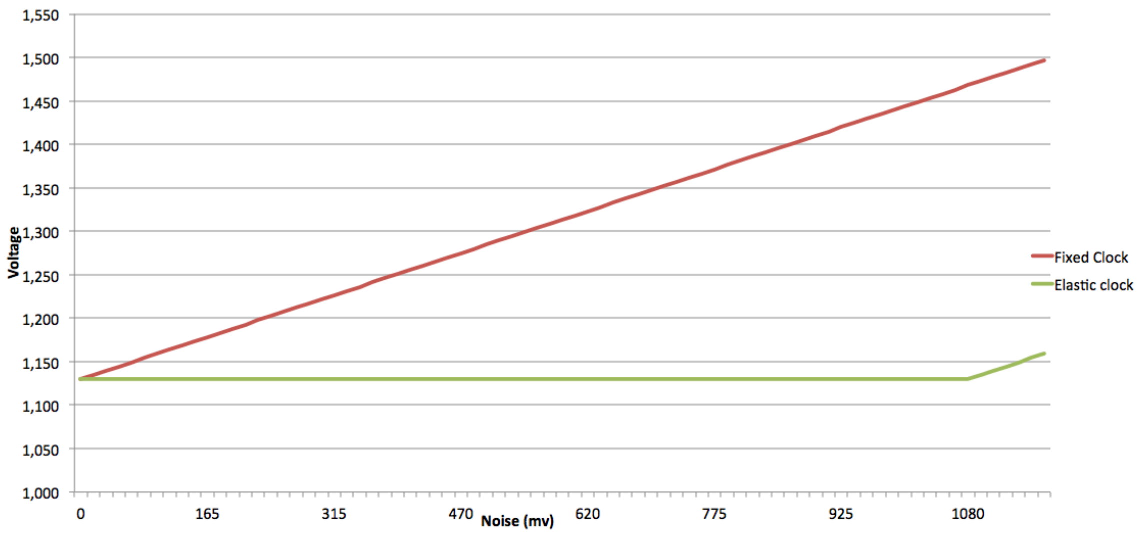


Figure 5.5: Tolerance over noise using elastic clocks vs a fixed clock. Both running at 50MHz.

As expected, the failure line progression of the fixed clock is lineal over the noise induced to the input. This is what it was expected to happened since the noise increases the critical path while the clock speed remains the same. Actually, this depends on the frequency of the noise: a frequency larger than the clock frequency will produce different result because within the same cycle peak to peak voltages are experienced which means the critical path is balanced to its noise free value, even though a very high frequency can produce failures too. These errors are produced due to the clock distribution tree delays but this is beyond the scope of this project. These last two phenomena could not be proven because the signal generator allows a 20MHz maximum frequency.

Elastic clocks has behaved as expected. Its tolerance to the noise is higher. Actually, the circuit does not even fail until the noise is increased to 1,130V peak to peak. At this point, the circuit is near the technology threshold, which is 0,6V and to be under it will make the board to be deprogrammed, and near the threshold the circuit becomes more sensitive to the noise. Nevertheless, at this point the to lines, fixed and elastic clocks, become parallel because the elastic will need to increase the voltage to support higher levels of noise.

The chart in Figure 5.5 shows exactly what is the problem of the synchronous circuits. In order to support a given level of noise it needs a safety margin in order to avoid failures due to timing violations because. The chart also shows how the elastic clock does not need this margin which could yield a diminution of the power consumption if this advantage is exploited.

As stated in previous chapters, the main reason of this high tolerance to noise of elastic clocks is that variability is very similar throughout the chip. Accordingly, the elastic clock and the logic suffers the same variations in throughput and the period is enlarged at the same time and same proportion than the critical path.

Now imagine that we need to design the circuit used for tests in order to support noise levels up to 1,080V peak to peak. If we decide to use the synchronous implementation using a fixed clock of 50MHz it would need to work at 1,468V in order to avoid failures. However, with the elastic clock implementation the circuit could still work at 1,13V which will lead in a reduction of the power consumption. Since the fixed clock works at 50MHz and the elastic clock works at 50MHz on average, reduction in power consumption can be directly translated to a reduction in energy consumption. The exact power reduction would be:

- Power consumption of the synchronous design, working at 1.468V with a current of 41mA:

$$P_{sync} = 1,468 * 0,041 = 0,064592W = 60,188mW$$

- Power consumption of the elastic clock design, working at 1,13V with a current of 30mA:

$$P_{elastic} = 1,13 * 0,030 = 0,0339 = 33,9mW$$

This leads to a reduction of consumption of 43,7% in terms of power and energy.

5.2.5 Immeasurable parameters

There are other parameters where asynchronous circuits have advantage over their synchronous counterparts. However, not everything is measurable within an FPGA and there are even parameters difficult to measure in ASICs, like manufacturing variability, which is measured in terms of percentage testing a large number of chips of the same type and wafer, which is impossible to do in the context of this project.

It would have been desirable to measure the temperature variability but the FPGA have no temperature sensors, which would make possible to monitor the temperature in order to do its corresponding analysis.

Chapter 6

Infinite Impulse Response system through A/D converter

Until now, the whole project has been tested with a dummy circuit with testing as its only purpose. In this chapter, a real case of use is presented: an IIR – *Infinite Impulse Response* – system, which takes data from an analog to digital converter.

The closed-loop feedback presented in this section is the same as the one presented in Chapter 4 where the generic closed-loop is presented but in this case the data producer circuit has been substituted by an ADC and the data consumer circuit by the IIR. The actual aspect is shown in Figure 6.1. A signal generator is used in order to induce different amounts of works to the IIR circuit.

During this section, an introduction to the IIR system is firstly done and later the ADC and its protocol are briefly explained. Finally, the closed-loop feedback is tested the same way the RISC processor was tested in section 6 but since the whole system is working the voltage scaling is done automatically to obtain the reference clock or the optimum voltage for a given workload.

6.1 Infinite Impulse Response system

Infinite Impulse Response is a property of signal processing systems and systems with this property are known as IIR systems. IIR systems have an impulse response function that is non-zero over an infinite length of time.

The function time of a single input depends on how it varies over the previous.

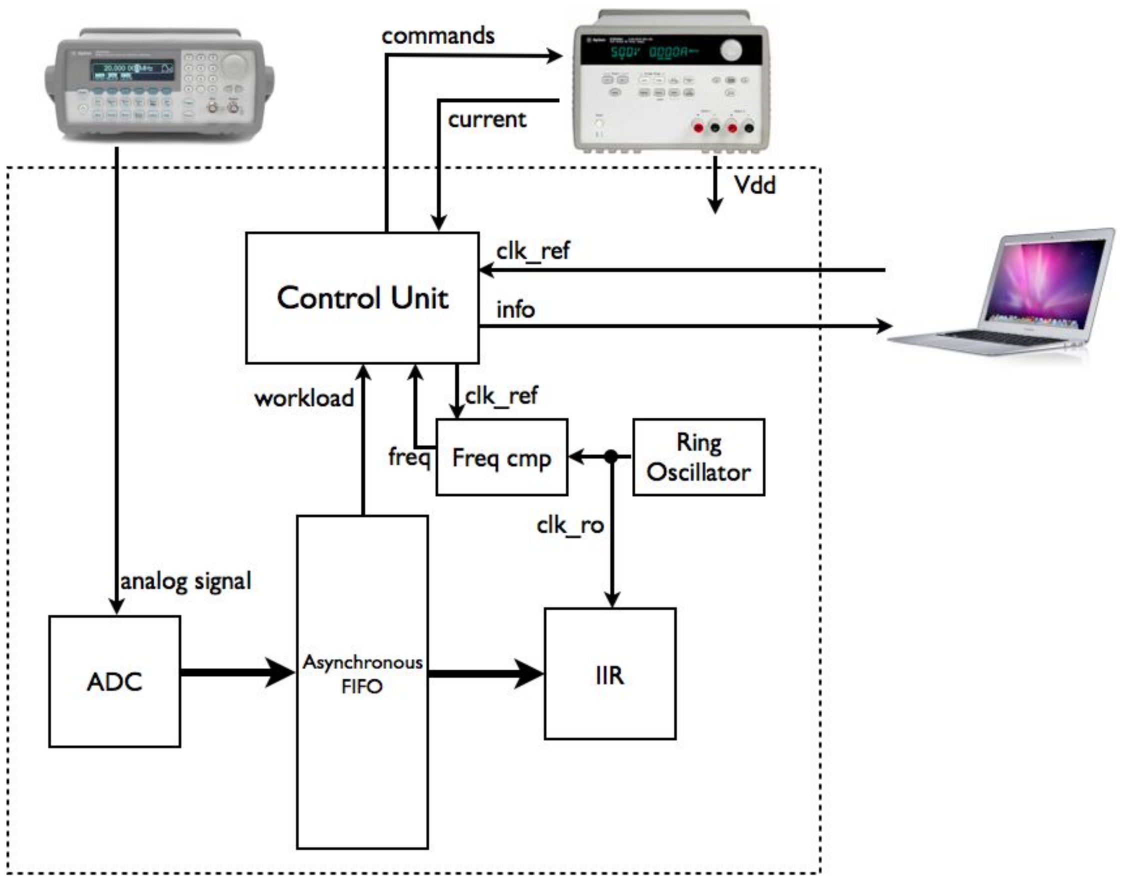


Figure 6.1: Block diagram of the closed-loop feedback implementing the ADC and the IIR.

In digital IIR filters, the output is immediately apparent in the equations defining the output, i.e., there is a feedback between the output and the input.

Figure 6.2 shows the block diagram of a simple IIR filter.

The IIR system used performs a low-pass filter. A low-pass filter is an electronic filter that passes low-frequency signals but attenuates signals with frequencies higher than the cut-off frequency.

The aim of the project was not to develop a DSP – *Digital Signal Processor* – system like the IIR filter. For this reason, the module has been obtained from the project opencores.org [2], the world's largest open source hardware community. The module is developed in VHDL instead of Verilog like the rest of the project. However, a wrapper was needed to make as interface between the VHDL subsystem and the rest of the Verilog design along with some minor changes to adapt it to the needs of the project.

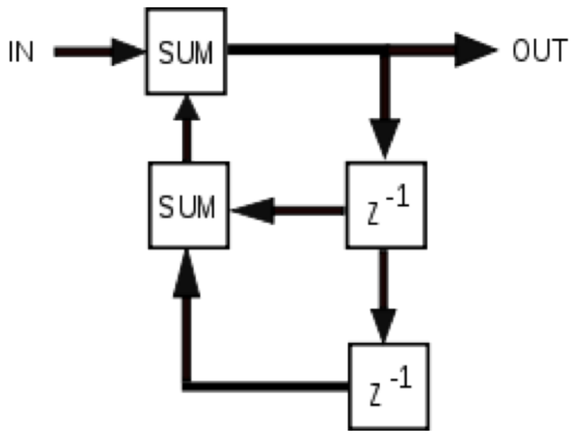


Figure 6.2: Block diagram of a simple IIR filter system. source: http://en.wikipedia.org/wiki/Infinite_impulse_response.

6.2 Digital outputs from analog inputs

The IIR filter is one example of a DSP system and this kind of applications need of signals in order to work as they are intended to work. For this reason, there was needed to input data from signals. Observing the Spartan 3E board specifications the only way to do that was using the analog to digital converter that is installed inside the board.
Before presenting the devices used for the conversion of analog signals to digital data, SPI bus has to be presented since it is the bus used for the communication between the FPGA and peripheral devices such as the ADC or the amplifier, which has been also used.

6.2.1 Serial Peripheral Interface Bus

The SPI bus is a synchronous serial data link standard, named by Motorola, that operates in full duplex mode. Devices using SPI communicate in master/slave mode. Multiple slave devices are allowed with individual slave select lines. Figure 6.3 shows the common connection between master and slaves.

The SPI bus specifies four logic signals:

- SCLK: serial clock – output from the master –.
- MOSI: master output, slave input – output from master –.
- MISO: master input, slave output – output from slave –.

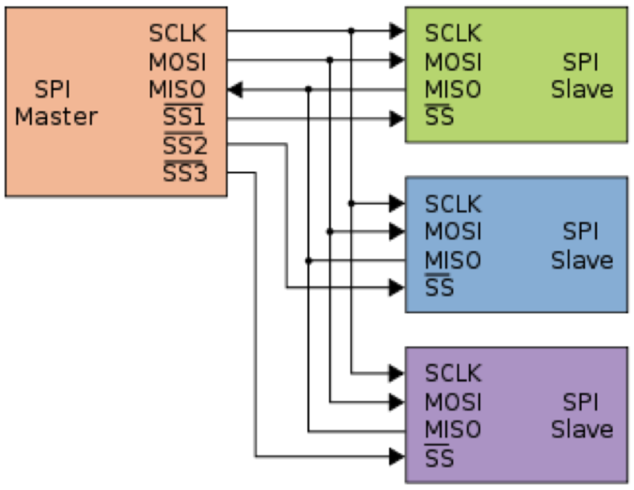


Figure 6.3: SPI Bus: master and three slaves. source: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus.

- SS: slave select – active low, output from master –.

To begin a communication, the master first configures the clock, using a frequency that is supported by the slave.

Then, the master switches the appropriate chip select bit for the desired chip to a logic 0 – because the chip select work with active low, i.e. its on state is asserted with a 0 –. Then, transmission can begin. During each SPI clock cycle: the master sends a bit through the MOSI line and the slave reads it from the same line and the slave sends a bit through the MISO line and the master reads it from the same line. Not the two transmissions are mandatory at each cycle, but since the protocol is full duplex, they can occur at the same cycle.

Note that the FPGA works always as the master and, hence, it is who generates the clock and all the control signals needed for transmitting.

6.3 Analog to Digital Converter

The board includes the LTC1407A-1 [14]. The LTC1407A-1 is a 14-bit, 3Msps – millions of samples per second – ADC with two 1.5Msps simultaneously sampled differential inputs. The devices draw only 4.7mA from a single 3V supply, which means the device is a low-power device, perfect for our purpose.

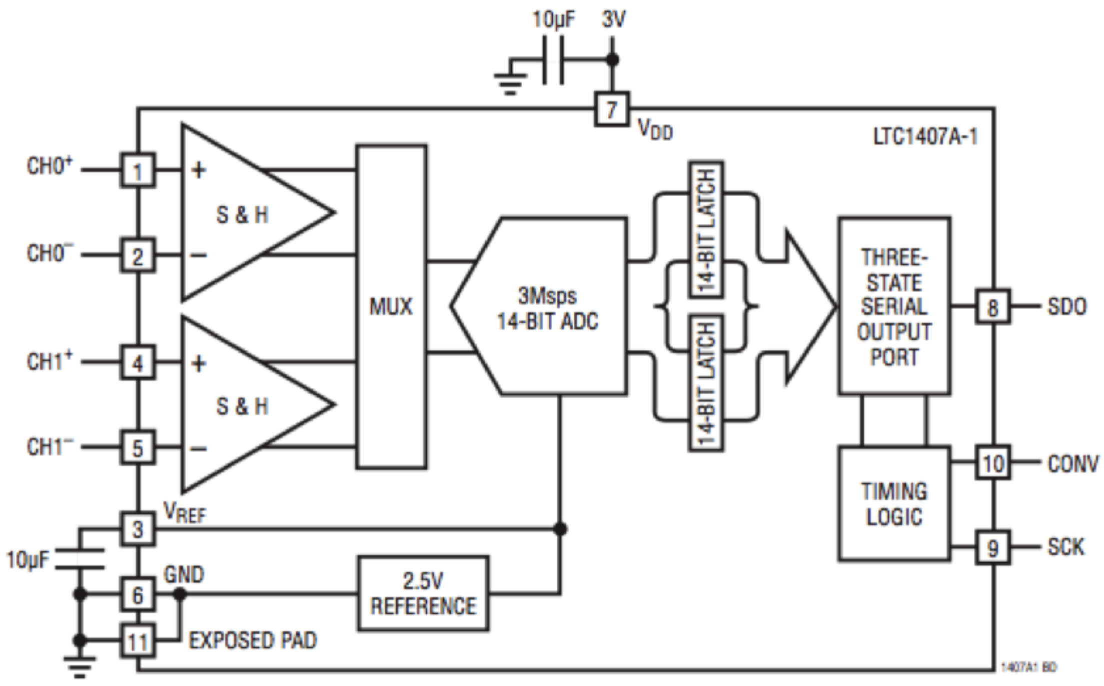
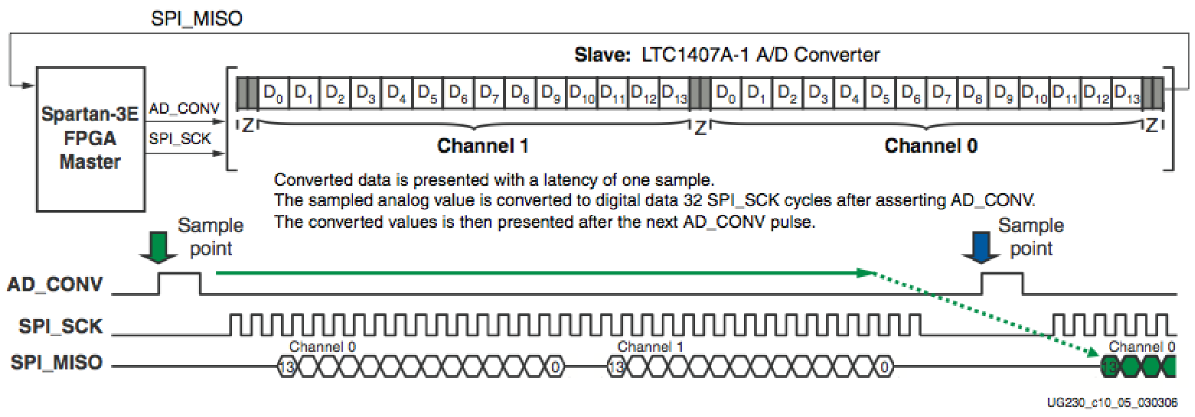


Figure 6.4: ADC block diagram.

Figure 6.4 shows the block diagram of the ADC. It shows the internal design. Two channels with their respective inputs, which pass through a MUX in order to select the correct input each time. The two channels works all the time but first the channel 0 is sampled and then the channel 1. The ADC outputs a 14-bits result, which is sent through the SPI bus.



Converted data is presented with a latency of one sample.
The sampled analog value is converted to digital data 32 SPI_SCK cycles after asserting AD_CONV.
The converted values is then presented after the next AD_CONV pulse.

Figure 6.5: Analog to digital conversion interface and timing.

Figure 6.5 shows detailed transaction timing. The AD_CONV signal is not a traditional SPI slave select enable. It is used to start the conversion and after it is set to low the ADC needs two SCK cycles to finish the conversion and start transmitting. After that it uses 14 cycles to send the conversion data of the channel 0 and 14 more for the channel 1 with two cycles between transmissions and two more at the end. With all, it takes 34 cycles to finish the communication sequence.

6.4 Programmable Pre-Amplifier

The board also includes a programmable pre-amplifier LTC6912-1 [15], which provides two independent inverting amplifiers with programmable gain. The purpose of the amplifier is to scale the incoming voltage on the two inputs so that it maximizes the conversion range of the ADC, namely 1.65 ± 1.25V.

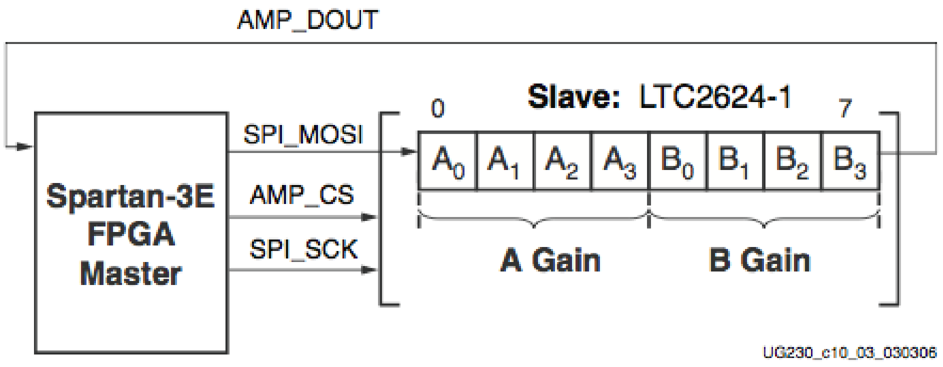


Figure 6.6: SPI Serial Interface to the amplifier.

Figure 6.6 shows the interface of the SPI to the amplifier. The gain is sent through the SPI_MOSI line from the FPGA and it is coded in 4-bits for each channel, which is a total of 8-bits. AMP_CS is used as chip select. AMP_OUT echoes the SPI_MOSI and it can be ignored in most applications.

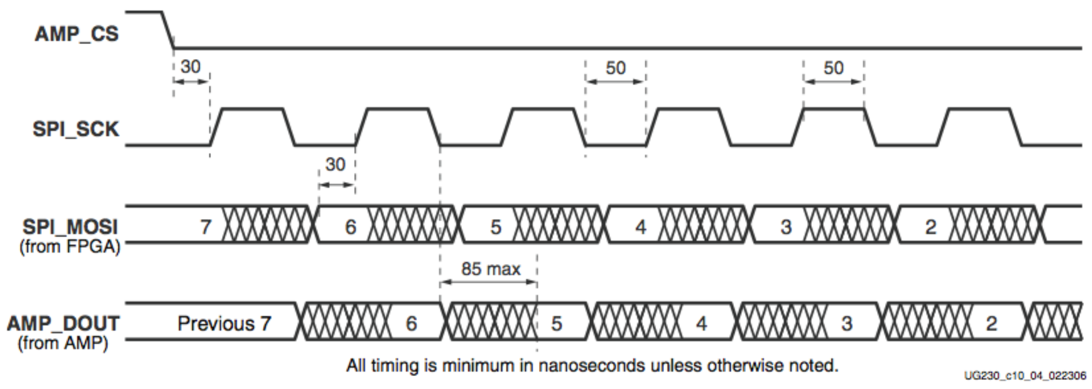


Figure 6.7: SPI Timing when communicating with the amplifier.

Figure 6.7 shows the timing of the communication with the amplifier. Once the AMP_CS is set to low, the amplifier starts to listen what goes trough the MISO line. Each SCK cycle a new bit is sent with LSB sent first.

The gain selected with the design is -1, which corresponds to the byte "00010001" and it gives a range of voltage between 0.4V and 2.9V.

6.5 Testing the real case

As has been done before while testing the elastic clock, in this section the whole closed-loop feedback will be tested. The design includes all the parts mentioned in the *Proof of Concept* section, which are the working subsystem and the monitoring subsystem.

The monitoring subsystem includes all the logic that communicates with the computer. The board sends to the computer information about workload, frequency and power consumption and it receives request from the computer in order to change its operating mode, which can be based on workload or a reference clock – also set by the computer –. The working subsystem includes the data generator, which is the ADC, governed by the fixed clock and the data consumer, which is the IIR, governed by the elastic clock. Between these two elements there is a crossing-domain element, which is the asynchronous FIFO previously mentioned.

In this case, it is not possible to reproduce the exact same tests as in Chapter 5 in which the processor was tested only including the elastic clock and varying the voltage manually. Now, the circuit includes the closed-feedback loop governs part of the circuit and the power scaling is done automatically by the control unit trying to adapt the input voltage and the oscillator to satisfy the needs of the circuit.

6.5.1 Workload mode

The closed-feedback loop, along with the elastic clock, provides a reduction of the power consumption and, hence, energy consumption. However, this reduction can be led by the simple fact of exploiting the properties of the elastic clock – smaller safety margins for the period – or it can be also led by the adaptive voltage scaling that provides the closed-feedback loop.

As stated before, the more the variation of the input signal, the more the job – clock cycles – has to do the IIR system. This is a perfect environment to test the closed-feedback loop and it has been exploited using the signal generator Agilent 33220A capable of generate sinusoidal waveform signals from 1Hz to 20MHz.

For the first test, we have tested the operating mode in which the adjustment is done to achieve the lowest power consumption given the workload at every moment – called work operating mode – while the input signal is being varied.

The chart on Figure 6.8 shows how the voltage auto adjusts with different input signal frequencies[1]. The maximum allowable value is 1.30V and the minimum is 0.75V. The maximum has been chosen for strictly safety reasons because it is not good for the board to work at a very high voltage. The minimum was chosen in order to let the monitoring subsystem work without failures – remember that it is working in synchronous mode with the 50MHz clock – and thus, be able to see all the information. However, the minimum voltage could be lower but always giving a safety margin to accept noise without reaching the transistor threshold, which would make the board to be deprogrammed – and it is 0.6V –.

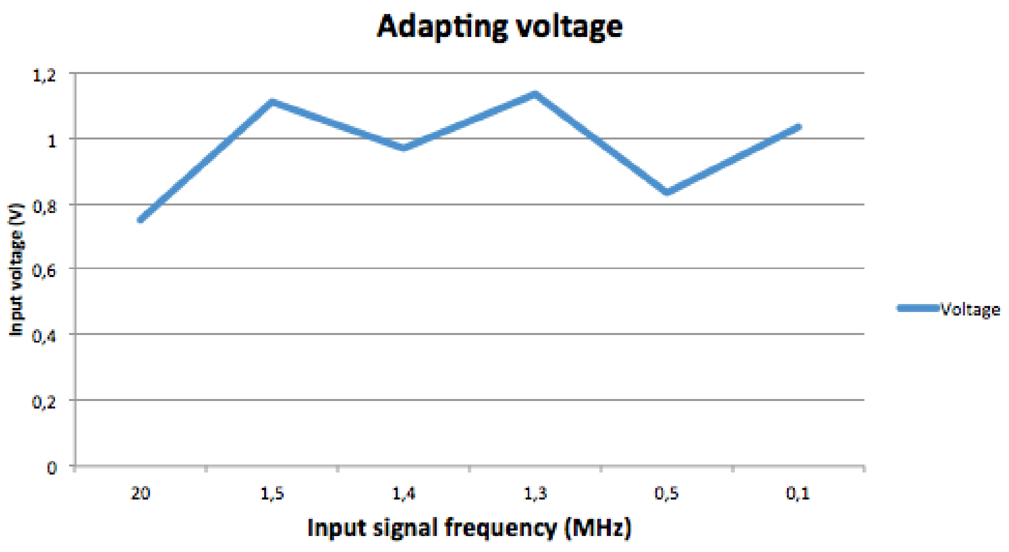


Figure 6.8: Adapting voltage depending on the workload.

[1]Note that more frequency does not mean more work. This is because the ADC converter cannot sample at the needed speed given by the Nyquist-Shannon sampling theorem, which states that the sample speed has to be at least the double of the signal frequency in order to reproduce without loss the waveform. Since the amplitude of the signal is always the same, higher frequency does not directly imply higher variations because it depends on how the ADC is sampling.

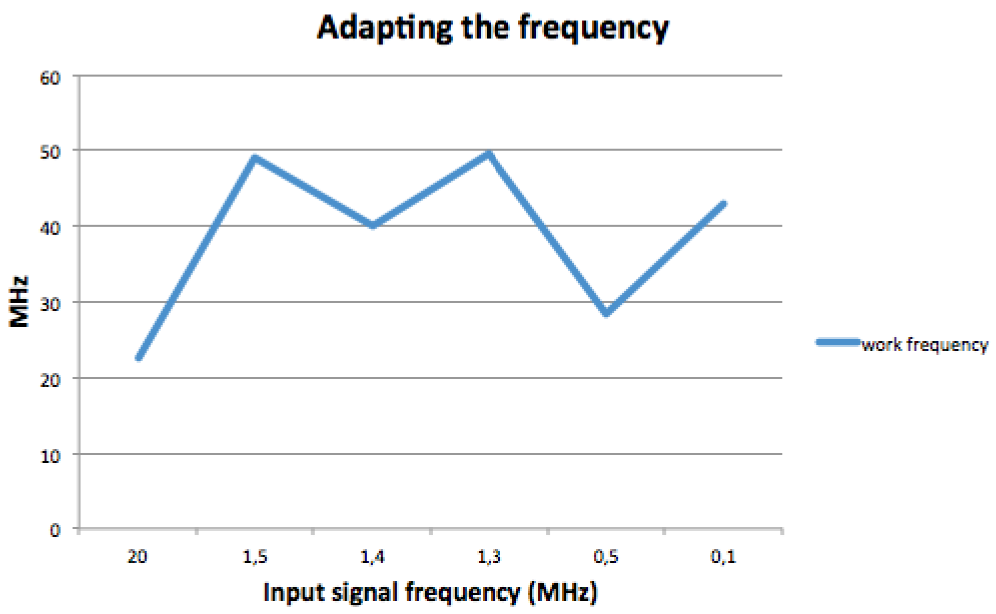


Figure 6.9: Adapting frequency depending on the workload.

Chart in Figure 6.9 shows the reason why it is interesting to scale the voltage with an elastic clock: the adjustment of the clock speed. A voltage scaling leads to a frequency scaling in a similar way. Comparing the two charts we can see how they are correlated at any time.

Chart in Figure 6.10 shows the results of the ultimate goal of the closed-feedback loop, which is power reduction. The reduction is calculated taking the consumption of the synchronous circuit and then comparing it with the consumption of the closed-feedback loop at any moment. As we can see, it has the same shape that the two previous charts but it is inverted because less voltage and less frequency imply more power savings.

Figure 6.11 shows an example of the interface when trying to adapt to the optimum voltage and frequency.

In the example shown in the Figure 6.11, the circuit was in a state of minimum power consumption because the workload was at its minimum level. Then, a 100KHz signal is induced and the workload gets maximum levels and the circuit increases its speed until a stable state is achieved again. As the figure shows, the circuit finds in 0.96V the optimum voltage level and it corresponds to 39.35MHz of clock frequency.

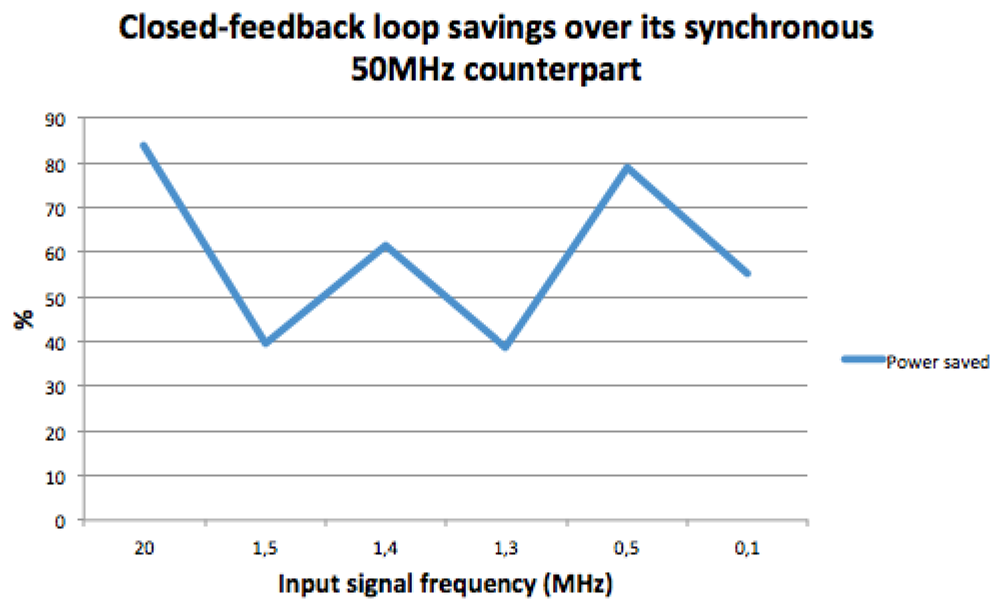


Figure 6.10: Closed-feedback loop savings over its asynchronous counterpart.

6.5.2 Reference clock mode

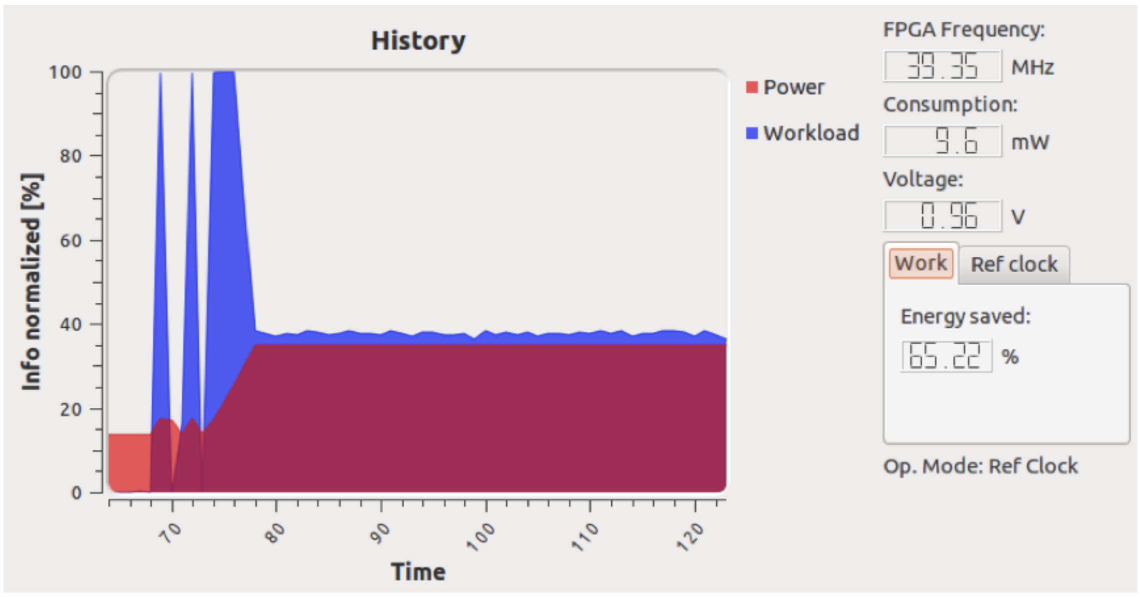
The circuit can also work at a mode in which there is a reference clock defined with the software interface and the closed-feedback loop tries to run at that speed varying voltage and adapting it to the variability.

The Figure 6.12 shows how the circuit tries to reach the speed of 50MHz. However, the precision of the supply voltage is too low for what the circuit needs and it gets a frequency of 49.72MHz instead of the 50MHz of the reference clock. This frequency is got by working at 1.12V, which corresponds to 17.92mW of power dissipation.

Even in this mode there the circuit is saving energy because it tries to achieve the speed with the minimum amount of power possible. In this case, comparing the synchronous at 50MHz and the asynchronous at roughly 50MHz, the closed-feedback loop is achieving a savings of the 35%. Since they running at same speed we can directly translate the savings in power to savings in energy.

6.6 Energy savings

Energy consumption is related to power consumption but a circuit consuming the half of the power of another taking the double of time to perform a task are con-



FPGA Frequency: 39.35 MHz

Consumption: 9.6 mW

Voltage: 0.96 V

Energy saved: 65.22 %

Op. Mode: Ref Clock

Figure 6.11: Interface of the monitoring software running in workload mode.

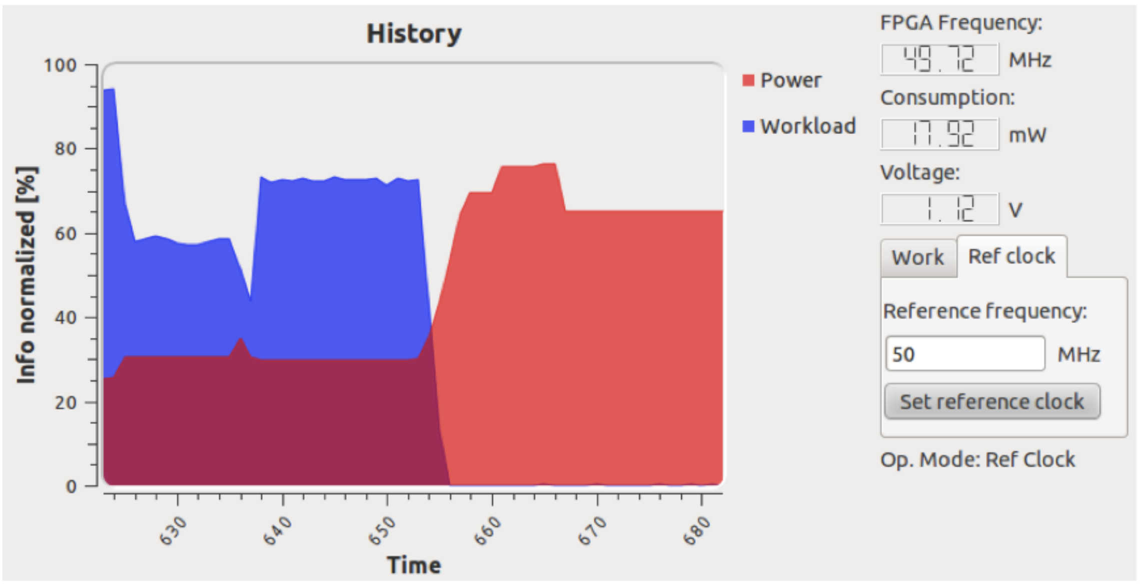
suming the same amount of energy. The idea of this project is to reduce energy consumption but there is no mention about it in the charts of the tests. Even though it is not important to compare instant information energy consumption will be discussed now.

Every time the circuit decides that it needs to vary the voltage it has to send a command to the power supply and the gap between commands needs to be of at least 180 milliseconds. Hence, for each command it takes the following amount of time, calculated in table 6.1:

```
Transmission speed = 9600 bauds = 9600 bits/s

Bits per byte = 8bits + 1 stop bit = 9bits

Length of a command:
VOLT UP\n = 6 character + 1 space + 1 new line = 8B
VOLT DOWN\n = 8 characters + 1 space + 1 new line = 10B

GAP = 180 milliseconds.
```

Table 6.1: Time between voltage's scalings.

With all of this we can calculate the exact amount of time the circuit needs



FPGA Frequency: 49.72 MHz
Consumption: 17.92 mW
Voltage: 1.12 V
Reference frequency: 50 MHz
Op. Mode: Ref Clock

Figure 6.12: Interface of the monitoring software running in reference clock mode.

to vary its voltage to achieve the optimum value. Now we take a test in which there is a signal input at 100KHz. Initially, the circuit is running at 1.2V and given the results it needs to downscale until 1.03V to achieve a frequency of 43MHz that seems optimum for the current workload. Hence, the board will need to send seventeen commands to the power supply in order to reduce the input voltage and this corresponds to:

$$Time = 17commands * (10\frac{bytes}{command} * 9\frac{bits}{byte} * \frac{1s}{9600bits} + 180ms * \frac{1s}{1000ms})$$

$$Time = 17 * 0,189375s = 3,219375s$$

The system takes 3,22s to achieve its optimum state and it is 0,19s at each step. With this information and knowing the consumption of the circuit at each voltage level – previously analysed –, if we make the circuit run during 5s the energy consumption will be:

- Synchronous circuit consumes 138mJ (milijoules).
- Asynchronous circuit with closed-loop feedback consumes 95,96mJ.

Hence, the energy reduction will be a 30,5%. In this case there is a reduction but we can assure the closed-feedback loop will never consume more than the synchronous if not necessary. The speed at 1.2V is fixed at 50MHz, which means that at same speed they consume the exact same amount of power. However, the closed-feedback loop is able to run at a higher frequency – and voltage –, which will imply higher power consumption, but since the power consumption – and voltage – is lineally related to the frequency – as seen in previous tests –, an increment on power consumption does not lead onto the same increment of energy consumption because the time to perform the same task will be reduced.

However, the maximum voltage can be limited in order to always consume less energy – same consumption at maximum speed but less consumption otherwise –.

Chapter 7

Project schedule and economic feasibility

7.1 Project schedule

For the development of the project some steps have been followed and they are briefly explained below.

- Project definition: at the start we did not know exactly what the project was about and we needed to define some general aspects.
- Study of existing approaches: once we knew the project was about to implement and test a proof of concept using asynchronous circuits was time to read bibliography for a better understanding of the field and also, to know more about some different approaches that could be implemented.
- Study of the development tools: The whole Xilinx suite and the FPGA world was something totally new and it was needed some time to get used to it.
- Implementing the elastic clock: after implementing some circuits for learning purposes was time to implement the cornerstone of the project and test it in some way.
- Implementing test circuits: once the elastic clock was running we needed something to test its benefits. We tried OpenRisc, MicroBlaze and some other until the final custom RISC processor was developed.

- Implementing closed-feedback loop: With the elastic clock running and completely tested was time to implement the closed-feedback loop, which is the biggest design in the project and included not only the elastic clock but also a whole communication system in order to transmit information to the computer and control the power supply through the serial port.

- Implementing the Linux monitoring application: The closed-feedback loop is synchronized with the computer and also controlled by it. This application works as the interface to show the information obtained from the board and to control some parameters of the board.

- Implementing ADC + IIR: Once everything was correctly working we needed a circuit with a real use to show and prove the improvement achieved by our approach in a real world case of use.

- Results evaluation: Thanks to the data obtained by the computer it was possible to obtain a lot of information about the behaviour of the circuit in many different environments. Once the data was collected was time to process it and take conclusions.

- Documentation: The writing of this document.

The planning through time is shown on Figure 7.1.

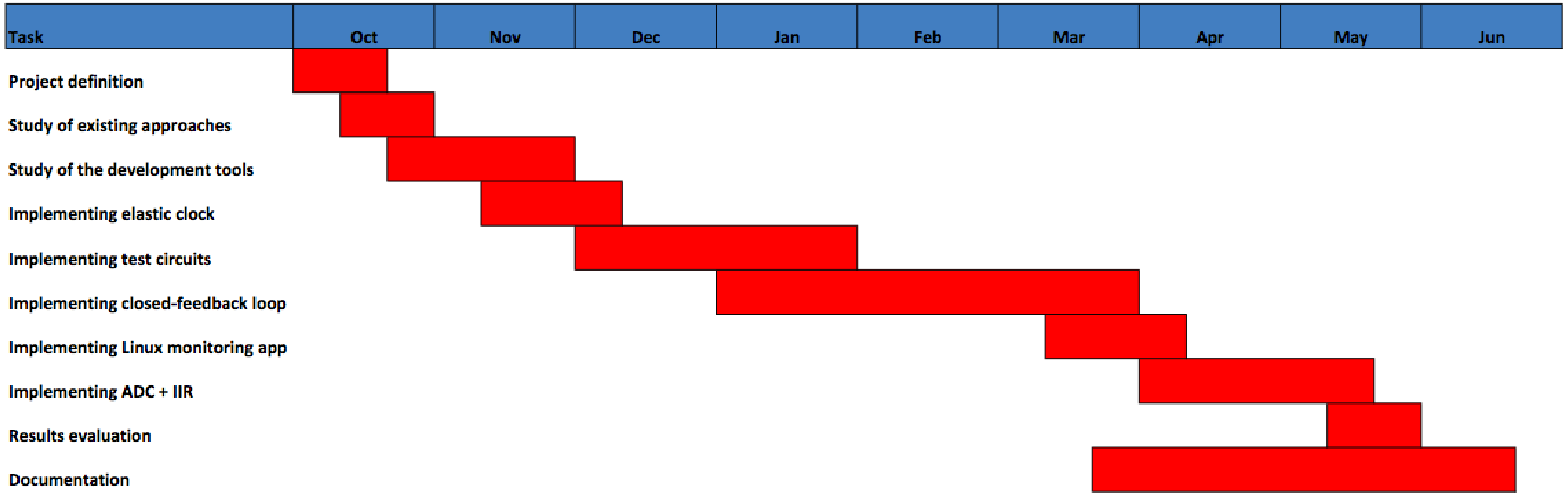


Figure 7.1: Schedule of the project.

7.2 Economic feasibility analysis

In this section is analyzed the economical costs around the project. There is an estimation of 1200 hours of work behind the project and they have been spread as follows:

- Architect: 300 hours
- Programmer: 500 hours
- Testers: 400 hours.

The estimation about the wages per hour is the following:

- Architect: 18€/hour
- Programmer: 12€/hour
- Testers: 8€/hour

Which give us a final personnel cost of:

$$Cost = 300h * 18\frac{e}{h} + 500h * 12\frac{e}{h} + 400h * 8\frac{e}{h} = 14600e$$

To this cost, we have to add the cost of the Xilinx license and the equipment used:

- Xilinx Design Suite Node-locked license: 2,995$ - 2400€
- Spartan 3E Starter Board: 189$ - 150€
- Programmable power supply Agilent E3646A: 840€

The total cost of the equipment and license is:

$$Cost = 2400e + 150e + 840e = 3390e$$

Hence, the total cost of the project is:

$$Totalcost = 14600e + 3390e = 17990e$$

Chapter 8

Conclusions

To conclude, the realization of the project has helped me to know and learn a field of informatics that was never, or very slightly, touched during the degree. It also showed how different are the software and hardware worlds basically because of the difficult of debugging, which has implied a lot of time during the realization of the project, but also because of the available information, very scarce.

On the other hand, the election of asynchronous circuits and, more specifically, elastic clocks was not the reduction of energy consumption per se but the reduction using more reliable circuits. The aims of the project were to use asynchronous design paradigm in order to reduce the energy consumption while making more reliable circuits. We have successfully proved the reliability of the elastic clocks

Nevertheless, the election of asynchronous circuits and, more specifically, elastic clocks was not the reduction of energy consumption per se but the reduction using more reliable circuits, i.e., in order to make synchronous circuits more reliable they usually consume more power and reliability and power consumption becomes a trade off. However, we have presented more reliable circuits with even less power consumption.

The main purpose of the project was to reduce the power consumption of circuits. This was done using asynchronous circuits because not only provide improvements in energy consumption but also help to increase the reliability of the circuit.

With the tests has been demonstrated how elastic clocks provide greater tolerance to noise and variability. This is so since variability is similar throughout the chip and thus, as stated in previous chapters, the critical path and the circuit logic are increased in virtually the same proportion as the clock period. Thanks to this phenomenon, the safety margins added to the clock period in a synchronous design are not needed and the voltage can be decreased and accordingly the energy consumption would decrease.

Taking advantage of the benefits of elastic clocks, which adapts its frequency with the circuit variability, a closed-feedback loop have been developed where the speed is automatically controlled only varying the input voltage. Thus, the system is reducing even more energy consumption because decreases speed, voltage and the margins for supporting noise without failure. Finally, we could say that the project has successfully achieved its purpose.

8.1 Future work

There are still a lot of possible improvements to do. One of the problems of the implemented system is the need of using an external power supply, which is very slow and not very precise. An improvement could be selecting an FPGA with a variable power supply on board or to print the system into silicon along with the needed power supply. This would make the closed-feedback loop be more accurate and optimum.

Not having the way to know the internal temperature has been a lack on the tests of the project, which implied not having analysis of the behaviour with temperature variability. An improvement could be to find an FPGA with a temperature monitor, not so uncommon, and use it to complement the actual tests.

Bibliography

[1] Asic world - verilog tutorial. `http://www.asic-world.com/verilog/index.html`.

[2] Opencores. `http://opencores.org`.

[3] Qwt user's guide. `http://qwt.sourceforge.net/`.

[4] P.A. Beerel, R.O. Ozdag, and M. Ferretti. *A designer's guide to asynchronous VLSI*. Cambridge Univ Pr, 2010.

[5] J. Bhasker and R. Chadha. *Static timing analysis for nanometer designs: a practical approach*. Springer Verlag, 2009.

[6] A. Bink and R. York. Arm996hs: the first licensable, clockless 32-bit processor core. *Micro, IEEE*, 27(2):58–68, 2007.

[7] P.P. Chu. *FPGA prototyping by Verilog examples: Xilinx Spartan-3 version*. Wiley-Interscience, 2008.

[8] J. Cortadella, M. Kishinevsky, and B. Grundmann. Self: Specification and design of synchronous elastic circuits. In *TAU'06: Proceedings of the ACM/IEEE International Workshop on Timing Issues 2006*, 2006.

[9] J. Cortadella, L. Lavagno, D. Amiri, J. Casanova, C. Macián, F. Martorell, J.A. Moya, L. Necchi, D. Sokolov, and E. Tuncer. Narrowing the margins with elastic clocks. In *IC Design and Technology (ICICDT), 2010 IEEE International Conference on*, pages 146–150. IEEE, 2010.

[10] D. Kinniment. *Synchronization and arbitration in digital systems*. Wiley Online Library, 2007.

[11] L. Lavagno and A. Sangiovanni-Vincentelli. *Algorithms for synthesis and testing of asynchronous circuits*, volume 232. Springer, 1993.

[12] D.H. Linder and JH Harden. Phased logic: Supporting the synchronous design paradigm with delay-insensitive circuitry. *Computers, IEEE Transactions on*, 45(9):1031–1044, 1996.

[13] M. Stein and P. Works. Crossing the abyss: asynchronous signals in a synchronous world. *EDN-BOSTON THEN DENVER THEN HIGHLANDS RANCH CO-*, 48(16):59–70, 2003.

[14] Linear Technology. Ltc1407-1/ltc1407a-1 - serial 12-bit/14-bit, 3msps simultaneous sampling adcs with shutdown, 2008.

[15] Linear Technology. Ltc6912 - dual programmable gain amplifiers with serial digital interface, 2008.

[16] Inc. XILINX. Spartan-3 generation fpga user guide, 2008.

[17] Inc. XILINX. Spartan-3e fpga starter kit board user guide, 2008.

[18] Inc. XILINX. Xst user guide, 2008.

[19] Inc. XILINX. Ise simulator (isim) - in-depth tutorial, 2009.

[20] Inc. XILINX. Planahead user guide, 2009.

[21] Inc. XILINX. Spartan-3 libraries guide for hdl designs, 2009.

[22] Inc. XILINX. Constraints guide, 2010.

[23] Inc. XILINX. Synthesis and simulation design guide, 2011.

[24] Inc. XILINX. Timing constraints user guide, 2011.

Appendix A

Development tools

For this project, several development tools have been used. For the first part – the hardware design –, a Xilinx FPGA and the Xilinx ISE – design suite – have been used.

A.1 Xilinx

Xilinx is an American technology company founded in 1984. It is known for inventing the *Field Programmable Gate Array* (FGPA) and as the first semiconductor company with a fabless manufacturing model.

Along with FPGAs, Xilinx provides a full suite for developing and designing ASIC solutions. This suite provides tools for the entire design flow – synthesis, P&R, etc. –. Since the FPGA used is a Spartan-3E from Xilinx, their tools have been used.

XILINX®

Figure A.1: Xilinx logo.

A.2 Xilinx ISE

Xilinx ISE is a software tool developed and provided by Xilinx for synthesis and analysis of HDL. The tool allows the designer to get a bit stream from the initial RTL performing all the steps automatically: synthesis, mapping, place & route and the generation of the programming file. Each one is an individual step and provides different information about the design.

ISE can perform every step with Verilog or VHDL indistinctly. Both can be mixed in a single design. The only requisite is that each module has to be written in only one language. Then, if a module needs to be instantiated within another one written in a different language, the second has to instantiate it with an equivalent interface, i.e., if the instantiated module is in VHDL and has two inputs of one-bit width and an output of one-bit, the other module has to instantiate the module with the same name and same inputs and outputs but in Verilog. This is not an standard – each tool has its way to wrap the languages –, so more complex structures may not be well wrapped and some errors might appear. As for the code, ISE has similar features to other IDEs like syntax highlighting.

Figure A.2: Xilinx ISE logo.

A.2.1 Design flow

ISE can perform any step on the FPGA design flow. Every step generates different reports and gives different information. Errors are detected while performing the corresponding step, e.g. if a signal is not declared is detected in the synthesis step but when an input pin is unconnected, the mapping step is who will trigger the error.

To execute a step is enough with double-clicking its name, Figure A.3 shows the look and the names. Then, the step and every previous step will be executed. When the execution is finished a symbol beside the step's name will indicate the result: a cross in a red background means that an error occurred; an exclamation mark in a yellow background means that the execution finished with some warnings; a check in a green background means that the execution finished perfectly.
Each step has other substeps that can help adjusting or analysing some aspects of the design. For example, after the synthesis it is possible to view the RTL Schematic;

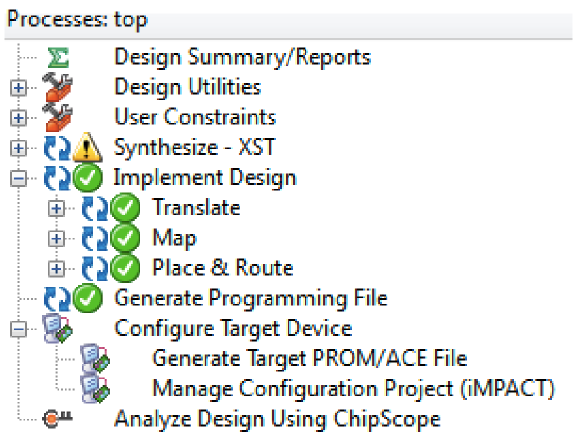


Figure A.3: Design flow of a project in Xilinx ISE.

after the mapping it is possible to generate a post-map static timing analysis; after the routing it is possible to view and edit the routed design.

A.2.2 Detailed reports

As stated before, reports are generated while the design advances. There is a report with general information with utilization of the FPGA resources but there are others with synthesis information – signals or modules trimmed during optimizations –, clock reports – information about the clock signals like fan-outs or skew – and also the timing analysis.

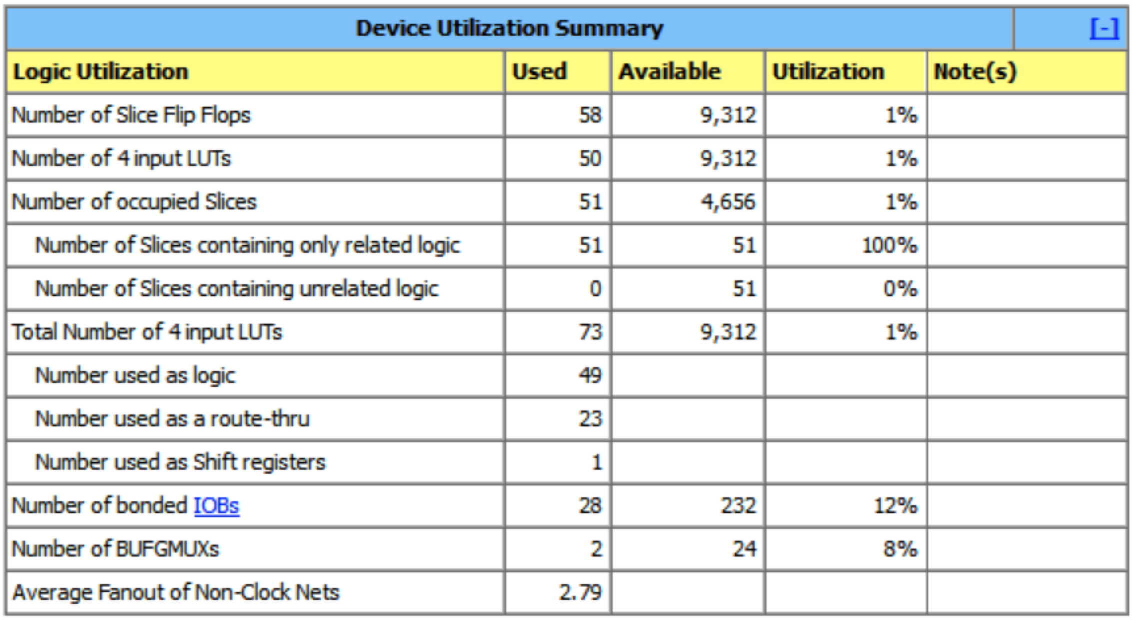
The timing analysis allows knowing exact information about the timing of the circuit like skew, slack or critical path within a clock domain. The analysis can be configured to be done at a given environment with parameters such as voltage, temperature or noise.

FIgure A.4 shows the look of the report of the utilization of the FPGA.

A.3 Simulation

Once a module is completed is time to test it. In hardware design this is done by simulation. ISE includes its own version of ModelSim Simulator. Figure A.5 shows the look of the simulator's interface.

The simulator allows the designer to test the circuit's signal response to different stimulus. Usually, a new and synthetic module is created. This module has no



Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	58	9,312	1%	
Number of 4 input LUTs	50	9,312	1%	
Number of occupied Slices	51	4,656	1%	
Number of Slices containing only related logic	51	51	100%	
Number of Slices containing unrelated logic	0	51	0%	
Total Number of 4 input LUTs	73	9,312	1%	
Number used as logic	49			
Number used as a route-thru	23			
Number used as Shift registers	1			
Number of bonded IOBs	28	232	12%	
Number of BUFGMUXs	2	24	8%	
Average Fanout of Non-Clock Nets	2.79			

Figure A.4: Device utilization summary.

inputs nor outputs and it instantiates the module under test. If necessary, the new module creates and initializes a clock signal, along with other signals needed by the unit under test.

Simulation is performed during a certain amount of time, usually a few microseconds or milliseconds.

However, simulation has several problems. The first one is performance. When the circuit gets complex simulation takes a long time to perform just several microseconds. The other problem is generating the input signals for the module under test. When it is a clock, a reset or any other predictable signal it is easy to generate it but when the design involve outer elements it is difficult to emulate these signals.

A.4 PlanAhead

This is a standalone program, even though usually launched from the Xilinx ISE. It gives information about mapping and placement. It shows the netlists and allows the designer to manually place elements throughout the board.

Figure A.6 shows the usual aspect of the main view of the program, which shows the board and the physical placement of its elements. When an element is selected, the software shows the netlists between it and the rest of the elements of

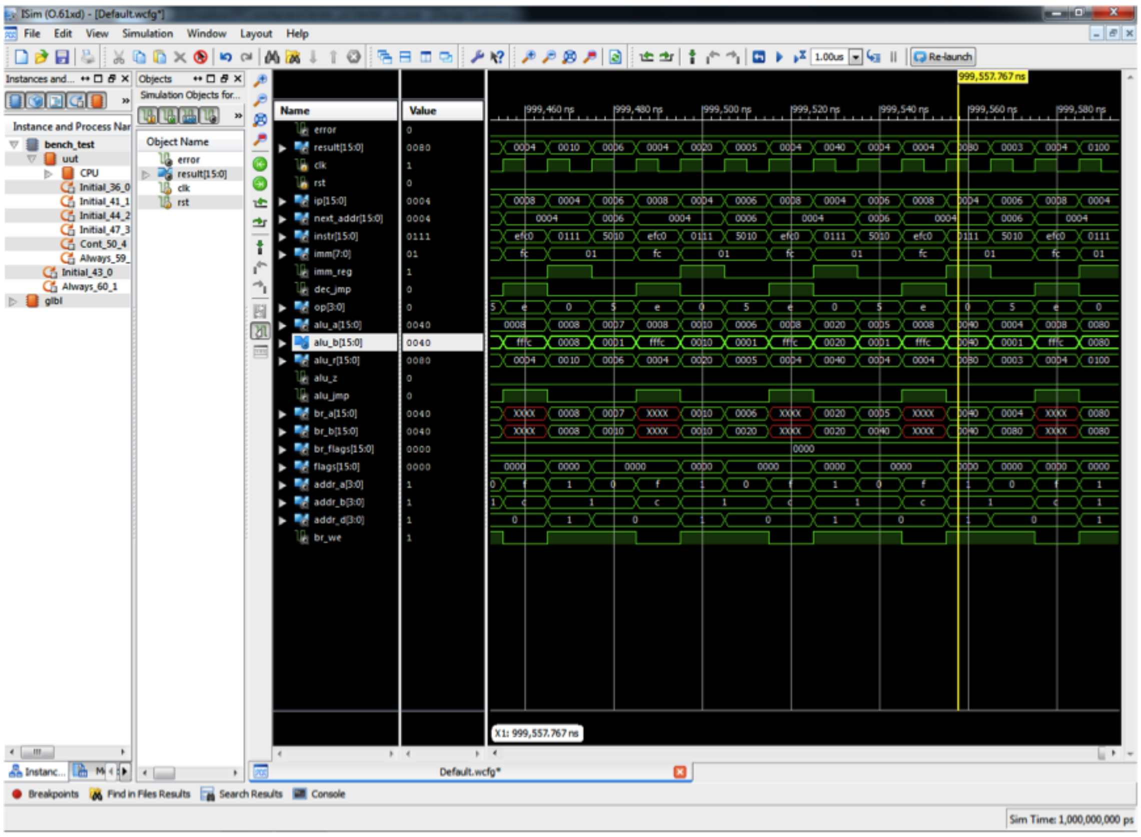


Figure A.5: ModelSim interface.

the circuit. The software also allows to drag and drop an individual primitive in order to manually place it and then a physical constraint is created, which means that the element will be locked to the chosen place.

Within the board, elements are divided into slices and each slice contains the same primitives. The Figure A.7 shows the aspect of four slices. The green elements are the elements that belong to the final circuit placement and the others do not contain any logic and are available.

A.5 Core Generator

The Core Generator is a software that allows the designer to use different IP – Intellectual Property – modules, which can be very helpful not having to implement design that are already available. For example, the Core Generator can provide a personalized FIFO, selecting parameters such as width or depth.

Figure A.8 shows the actual aspect of the software while generating a FIFO. Note the multiple possible signals that can be selected. Thanks to the Core Generator, the

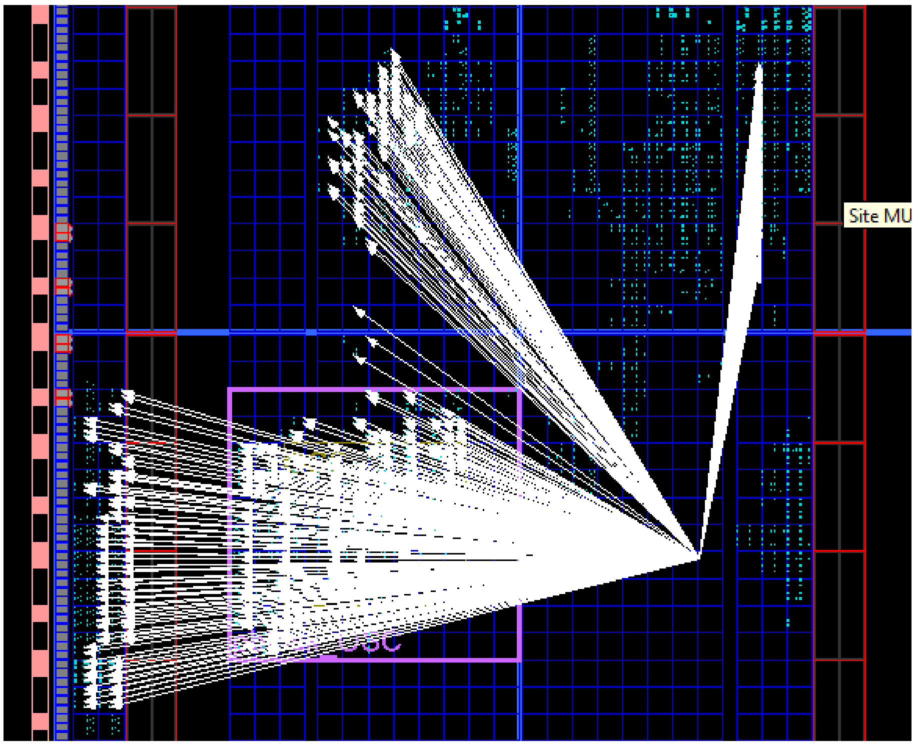


Figure A.6: PlanAhead showing connections between the output of the ring oscillator and the rest of the circuit.

personalization of the asynchronous FIFO used during this project for the closed-feedback loop has saved a lot of development time.

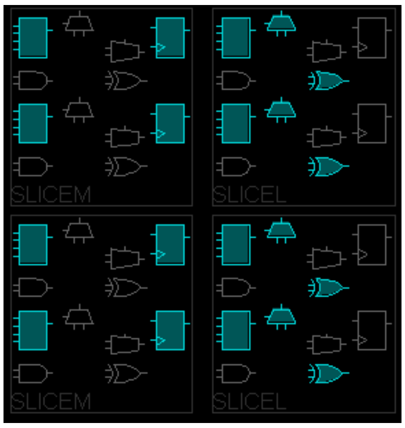
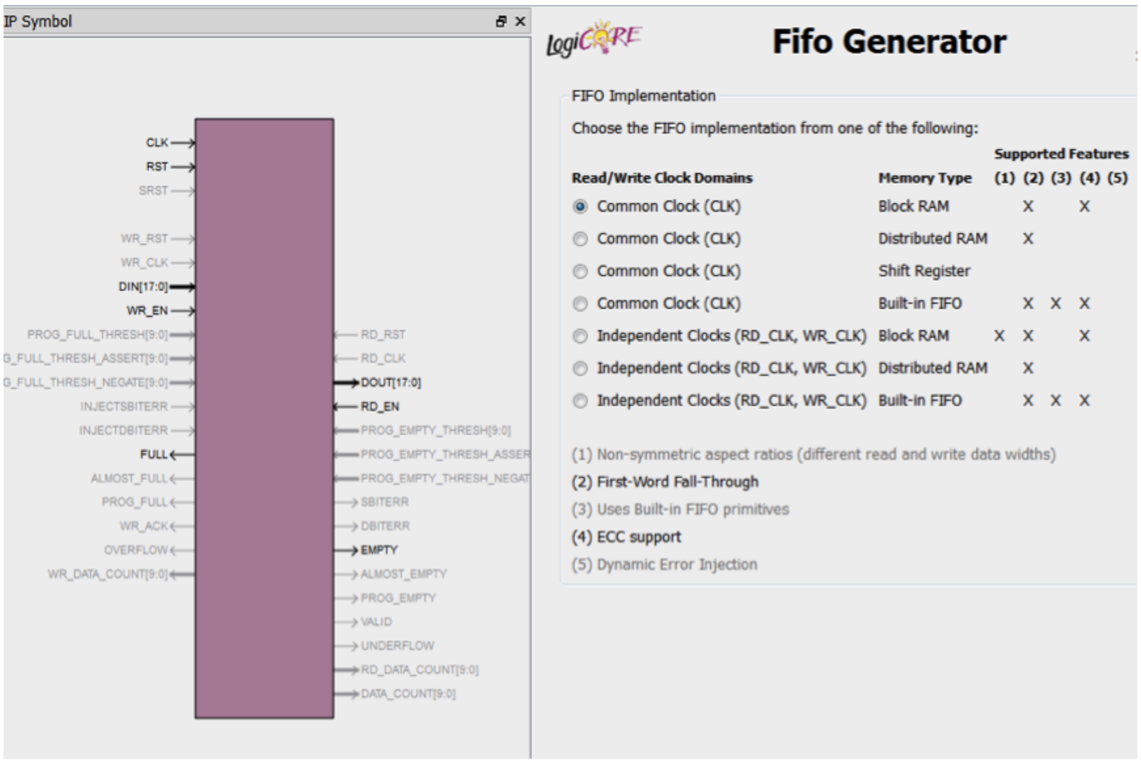


Figure A.7: Four slices of the FPGA and its occupancy.



Fifo Generator

FIFO Implementation

Choose the FIFO implementation from one of the following:

Read/Write Clock Domains	Memory Type	(1)	(2)	(3)	(4)	(5)
[✓ Common Clock (CLK)]	Block RAM		X		X	
Common Clock (CLK)	Distributed RAM		X			
Common Clock (CLK)	Shift Register					
Common Clock (CLK)	Built-in FIFO		X	X	X	
Independent Clocks (RD_CLK, WR_CLK)	Block RAM	X	X		X	
Independent Clocks (RD_CLK, WR_CLK)	Distributed RAM		X			
Independent Clocks (RD_CLK, WR_CLK)	Built-in FIFO		X	X	X	

Supported Features

(1) Non-symmetric aspect ratios (different read and write data widths)
(2) First-Word Fall-Through
(3) Uses Built-in FIFO primitives
(4) ECC support
(5) Dynamic Error Injection

Figure A.8: Four slices of the FPGA and its occupancy.

Appendix B

RS-232 Protocol implementation

The Spartan 3E includes two RS-232 ports but since it is an FPGA, it does not include the necessary logic to make the ports work. For this reason, it was necessary to study, understand and implement the protocol in Verilog in order to transmit through it.

RS-232 is a serial protocol used to transmit parallel data through a serial line between a DTE (Data Terminal Equipment) and a DCE (Data Circuit-terminating Equipment). The serial port signals are unidirectional and when connecting two devices, the input of one is connected to the output of the other. A line that is an output in a DTE is an input in a DCE device and vice-versa.

RS-232 includes a transmitter and a receiver. The transmitter is essentially a special shift register that loads data in parallel and then shifts it out bit by bit at a specific rate. The receiver, on the other hand, shifts data bit by bit and then reassembles the data.

If it is necessary to connect two DTE devices a special cable known as null-modem cable must be used.

The Spartan-3E has two serial ports: one DTE and one DCE. The computer is connected to the FPGA through its DCE port and the power supply through its DTE port. Since the power supply has also a DTE port, a null-modem cable must be used.

B.1 Voltage levels

The RS-232 specifies the voltage levels for the logical levels one and zero for the data transmission and the control signal lines. There are several signal levels defined in the standard and are used depending on the power supply available in the device, however, RS-232 transmitters or receivers must be able to withstand any voltage level up to ±25V. The most commonly used are ±5V, ±10V, ±12V and ±15V.
For data transmission, the logic one is defined as a negative voltage and the logic zero as a positive voltage. Figure B.1 shows the transmission of a character using voltage levels of ±12.

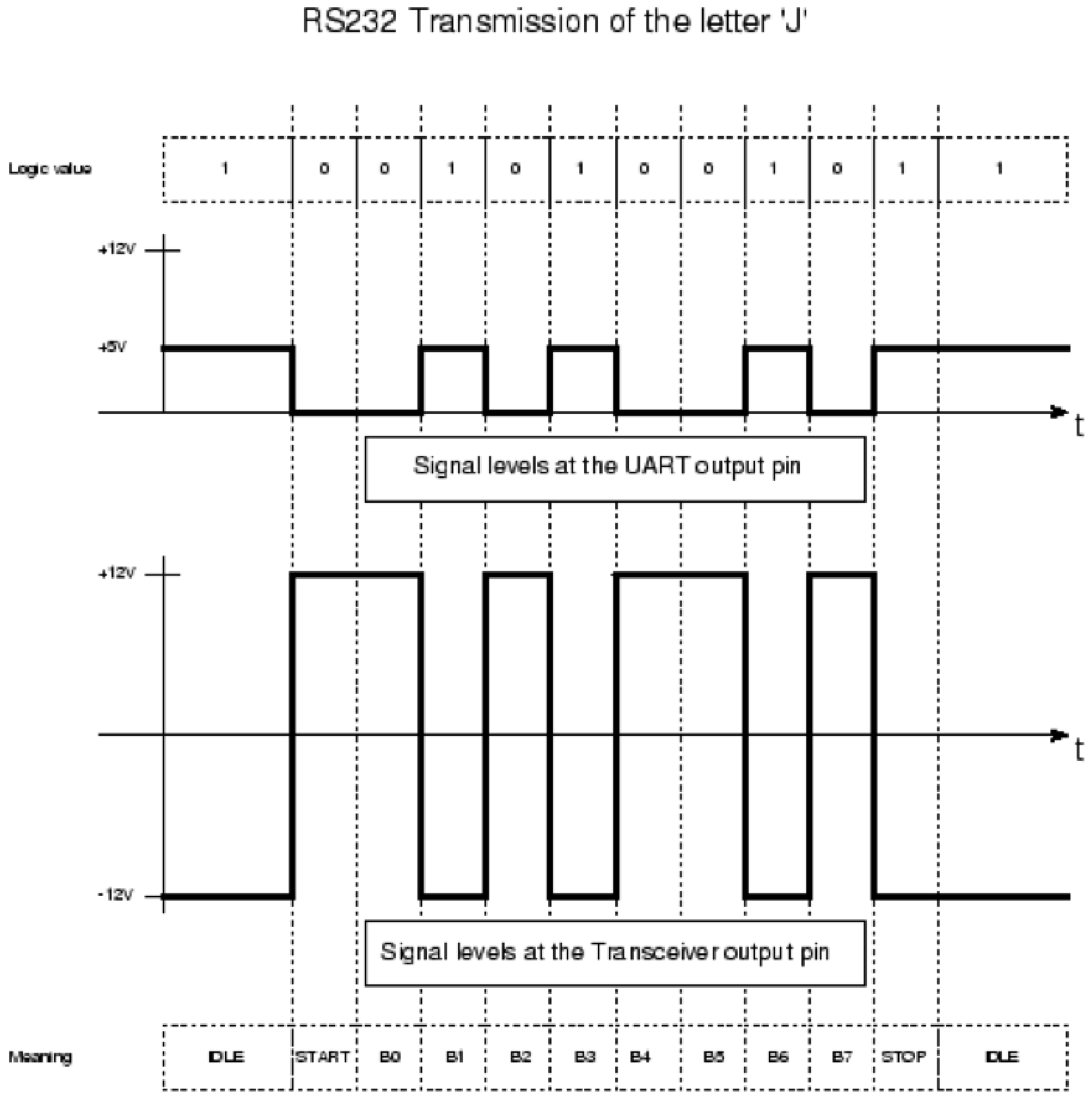

RS232 Transmission of the letter 'J'
Logic value
1
0
0
1
0
1
0
0
1
0
1
1
+12V
+5V
t
Signal levels at the UART output pin
+12V
-12V
t
Signal levels at the Transceiver output pin
Meaning
IDLE
START
B0
B1
B2
B3
B4
B5
B6
B7
STOP
IDLE

Figure B.1: Rs-232 Transmission of the letter 'J'. source: http://www.best-microcontroller-projects.com/how-rs232-works.html

Because the voltage level defined in RS-232 is different from the logic levels and, hence, the typical from that of FPGA I/O, a voltage converter chip is needed between a serial port and an FPGA's 110 pins. The board contains the necessary voltage converter chip and configures the various RS-232's control signals to automatically generate acknowledgment for the PC's serial port.

RS-232 is an asynchronous protocol as there is no clock transmitted at all. The standard defines some settings that are settled before transmitting; therefore, the two end-points have to set the exact same configuration. These settings are:

- Baud rate: the speed transmission. The standard does not define an exact speed but a serial of possible rates. The most commonly used baud rates are 2400, 4800, 9600 and 19200 bauds.

- Data bits: there will be seven or eight data bits with the LSB – Less Significant Bit – transmitted first. The option is given because the ASCII code needs only 7 bits for representing only characters. The use of 7 bits increases transmission speed when transmitting large blocks of data. If graphic symbols or raw data are needed to be sent, then 8 bits will be transmitted.

- Parity bit: the parity bit is a classic and simple error detection mechanism. It is possible to choose either odd parity or even parity or none at all, which will not send parity bit. This mechanism counts the number of data bits settled to 1. Then, if the odd parity is selected and the number of bits to 1 is odd, the parity bit will be 1, 0 otherwise. The even parity works in exactly the opposite way. However, if two bits fail, the parity mechanism fails.

- Stop bit: The stop bit is a period of time between the last data bit – or the parity bit if it is activated – and the start bit. The stop bit returns to the idle state before starting again. It is needed for synchronization purposes because at this point the two end-points know the transmission is finished. The stop bit can be set to 1, 1.5 or 2 bit periods.

The serial line is 1 when it is idle. Then, the transmission starts with the start bit, which is 0, followed by the data bits and the optional parity bit and finally ends with the stop bits, which are 1.

The Figure B.2 shows the transmission of a byte with 8 data bits, no parity and 1 stop bit. Note that the LSB of the data word is transmitted first.

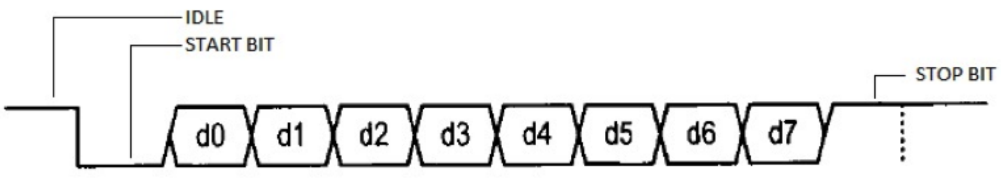


Figure B.2: Transmission of a byte in RS-232. source: http://www.best-microcontroller-projects.com/how-rs232-works.html

B.2 Implementing the RS-232 subsystem

The whole RS-232 subsystem, as shown in Figure B.3, is composed by four main components:

- RS-232 receiver: the circuit to obtain the data.
- RS-232 transmitter: the circuit to transmit the data.
- Baud rate generator: the circuit to generate the sampling ticks.
- Interface circuit: the circuit that provides a buffer and status between the RS-232 and the system that uses it.

B.3 Receiving subsystem

Since no clock information goes through the transmitted signal, the receiver needs to interpolate the information about the baud rate of the connection in order to know at which point it can sample for the data information. The technique used for this purpose is called oversampling scheme.

The *oversampling scheme* is used to estimate the middle point of the transmitted bits and then sample them at that point. The middle point is intended to be the point where the signal is more stable and also gives a margin if the two clocks are out of phase.

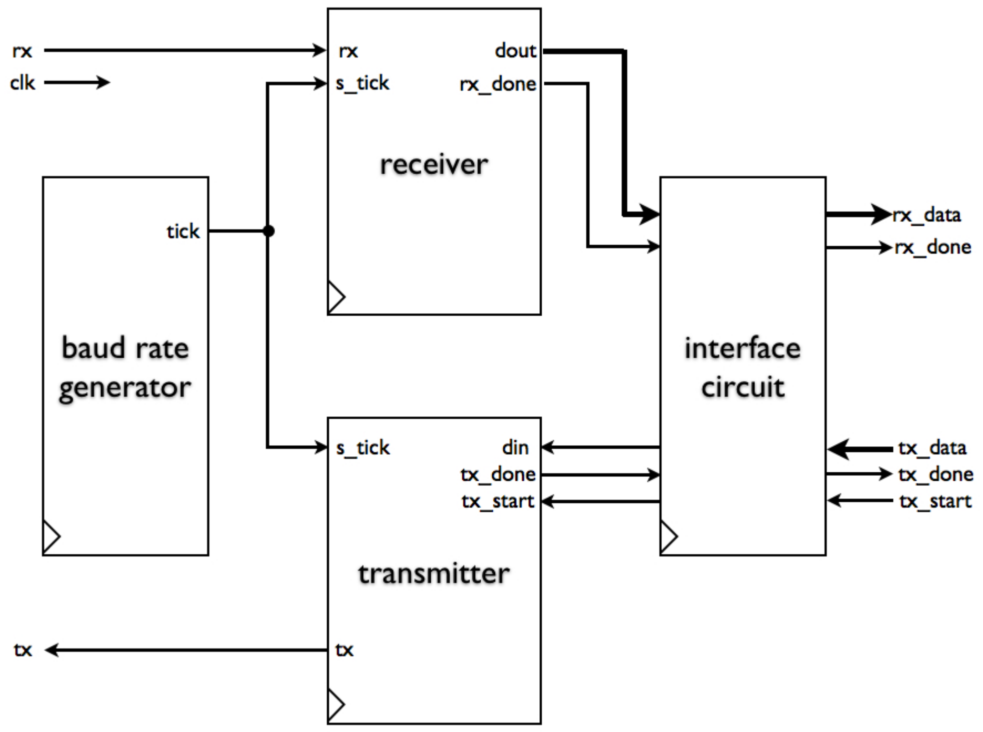


Figure B.3: Conceptual block diagram of the RS232 subsystem.

The most commonly used sampling rate is 16 times the baud rate, which means that each serial bit is sampled 16 times. When the start bit comes, starts the sampling tick counter. When the counter reaches 7 means that the signal is at its middle point, then the counter is set to 0 and restart. From here, every time the counter reaches 15, the value of the bit is retrieved and shifted into a register. The process is repeated until the stop bit comes, and then it starts from the beginning for a new byte.

B.4 Baud rate generator

The baud generator needs to generate a sampling signal whose frequency is exactly 16 times the RS232 baud rate. The sampling signal is used as an enable signal, so no new clock domain is created – avoiding clock-domain crossing elements –.

In this project, the baud rate utilized has been 9600 bauds. Thus, the sampling rate is 153600 – i.e. 9600 * 16 – ticks per second. Since, the Spartan 3E has a

50MHz clock, the baud rate generator has to generate a one-clock-cycle sampling tick every 326 clock cycles. This number comes from:

$$\frac{50 * 10^6}{153600} = 325,52$$

which means the generator needs a mod-326 counter.

B.5 Transmitting subsystem

The transmitting subsystem works in a similar manner to the receiving subsystem. Basically, the transmitting subsystem is a shit register that shifts out data bits at a given baud rate. The same baud rate generator controls the rate of the transmission but because there is no oversampling, the frequency is 16 times slower than that of the receiver.

The transmitter follows the protocol and is responsible of setting the signals at its corresponding value. For example, if it is not transmitting, the signal has to be in the idle state, then the start bit, the data bits, the optional parity and the stop bits. The duration of the bits is controlled by de baud rate generator but changes in the signal are made every 16 ticks of the baud rate generator – because of a 16 times slower frequency of the transmitter –.

B.6 Interface circuit

This circuit is just an interface to make possible to encapsulate the whole RS-232 subsystem into a bigger system. It only instantiates the subsystems and connects every signal with its correct port.