الجمهـوريـــة الجزائـريـــة الديمقـراطيـــة الشـعبيـــة
République Algérienne Démocratique et Populaire
وزارة التـــعلـيـــم العـــالـــي والبـحـــث العـلمـــي
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
جامعة وهران 2 محمد بن أ حمد
Université d'Oran 2 Mohamed Ben Ahmed

معهد الصيانة و الأمن الصناعي
Institut de Maintenance et de Sécurité Industrielle

Département De Maintenance en Instrumentation

MÉMOIRE

Pour l'obtention du diplôme de Master

Filière : Génie Industriel

Spécialité : Maintenance, Fiabilité, Qualité

Thème

Conception et réalisation d'un système d'étiquetage automatique par FPGA

Présenté et soutenu publiquement par :

Hadoud Ayoub **Et** **Saada Mohammed Laarbi**

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
ALDJOUA Abdelaziz	MCA	IMSI-Université D'Oran 2	**Président**
TITAH Mawloud	MCB	IMSI-Université D'Oran 2	**Encadreur**
BACHIR BOUIDJRA Bachir	MCB	IMSI-Université D'Oran 2	**Examinateur**

Année 2023/2024

Content

Abbreviations list.....3

I. Industrial Automation.....4

1. Introduction.....5
2. Definition.....5
3. History.....5
4. Modern Automated System Structure.....6

II. FPGA.....9

1. Introduction.....10
2. Definition.....10
3. FPGA Board Components.....11
4. Configuration Methods.....15
5. Uses and Benefits of using FPGA.....21

III. Binary logic Schematic design.....24

1. Introduction.....25
2. Binary System.....26
3. Logic gates.....27
4. Flip-Flops.....33
5. Schematic Entry Implementation.....37

IV. Realization Auto Labeling System.....50

1. Project Preparation.....51
2. Sketching.....52
3. 3D Modeling.....53
4. Electrical Circuit.....54
5. Schematic Entry Circuit.....56

Conclusion.....57

References.....57

Abbreviations list

FPGA: Field Programmable Gate Array

SoC: System on Chip

ADC: Analog to Digital Converter

DAC: Digital to Analog Converter

I2C: Inter-Integrated Circuit

SDA: Serial Data Line

SCL: Serial Clock line

AC: Alternative Current

DC: Direct Current

SMD: Surface Mount Device

MCU: Microcontroller Unit

IOs: Inputs & Outputs

CLBs: Configurable Logic Blocks

LUT: Look Up Table

MUX: Multiplexer

JTAG: Joint Test Action Group Port

USB: Universal Serial Bus

SRAM: Static Random-Access Memory

RTC: Real Time Clock

PLL: Phase-Locked Loops

VHDL: Verilog Hardware Descriptive Language

PLD: Programmable Logic Device

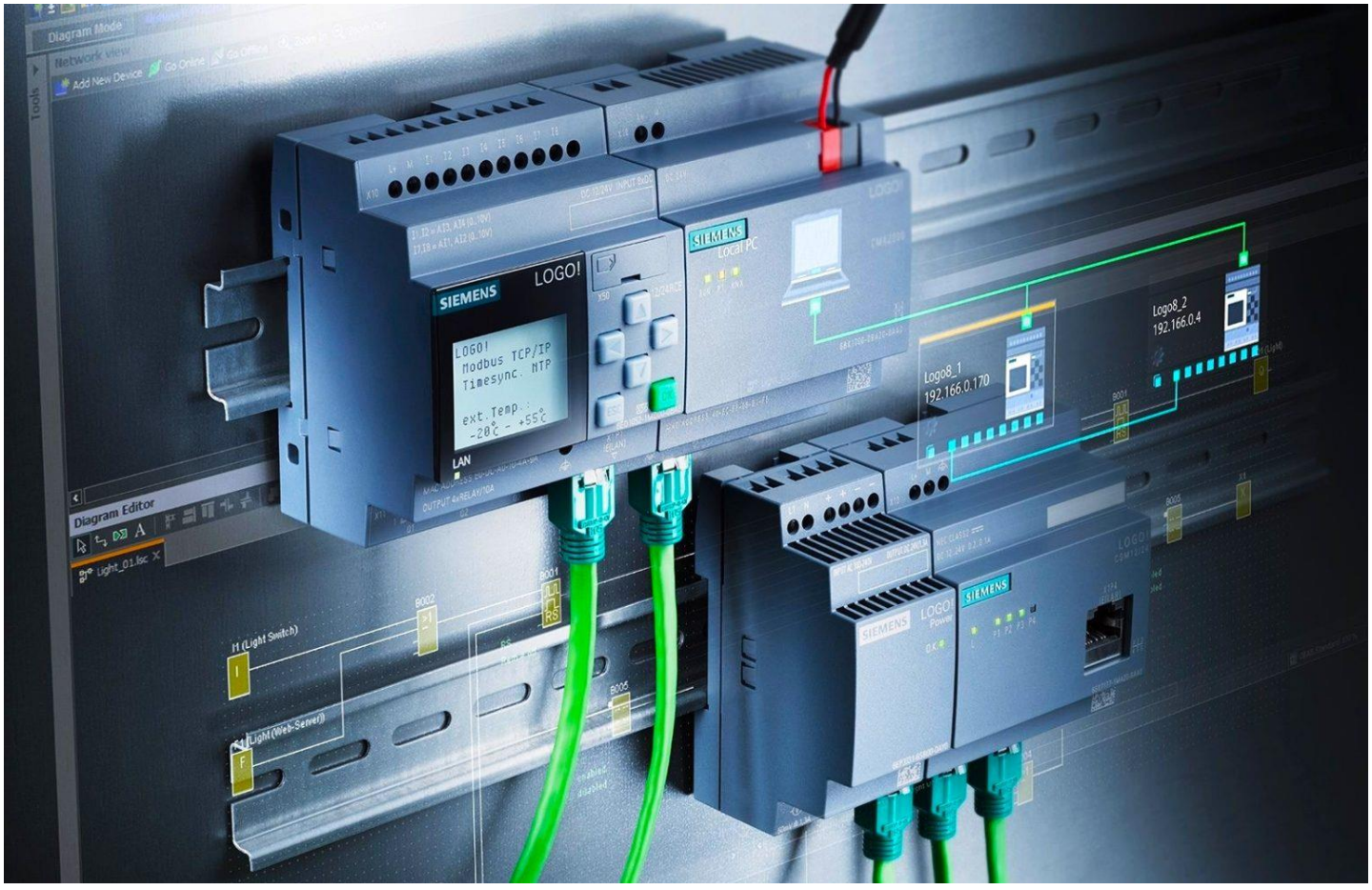
PWM: Pulse Width Modulation

ASICs: Application-Specific Integrated Circuits

CLK: Clock Signal

IR: Infra-RED Sensor

Industrial Automation



1] Introduction

The goal of every company is to achieve the greatest amount of profits by raising productivity, reducing human errors and improving accuracy. This is not possible if production depends on the old and cheap manual method, which lacks these conditions, despite the fact that humans are capable of handling complex tasks due to their possession of perception and awareness. But this is not enough with the large consumer demand, and to meet this need it was necessary to adopt another approach.....

2] Definition: What Is an Automated System?

An automated system is a medium that brings together control systems, sensors and actuating devices into a unit capable of performing tasks without complete human intervention. By that it reduce industrial risks, also guaranties higher levels of accuracy, productivity and quality.

These systems exhibit adaptability and versatility, allowing them to execute predefined tasks with precision and consistency while also accommodating dynamic changes in the environment or task requirements, and through continuous innovation and integration of emerging technologies such as artificial intelligence and machine learning, automation systems are poised to revolutionize industry practices and pave the way for a more interconnected, intelligent, and automated future.

3] History:

In the sense of using machines or equipment, automation dates back to the 11th century when miners used waterwheels to drain out water from underground tunnels and shafts. The modern form of automation took shape during the Industrial Revolution in the 1800s when automated processes and tools were used to increase factory productivity. Use of electricity in the1920s led to faster production process at the factory changing the factory floor dynamics. The application of feedback controllers by the industry during the 1930s and 40s was a significant step towards modern automation in manufacturing.

By 1980s the world saw new levels of automation with many sectors from manufacturing and retail to pharmaceutical and consumer goods embracing some or the other form of technology to further productivity.

4] Modern Automated System Structure:

In order to make a fully functioning hands-free automated system requires a solid connection between a specific components that we can break into two categories:

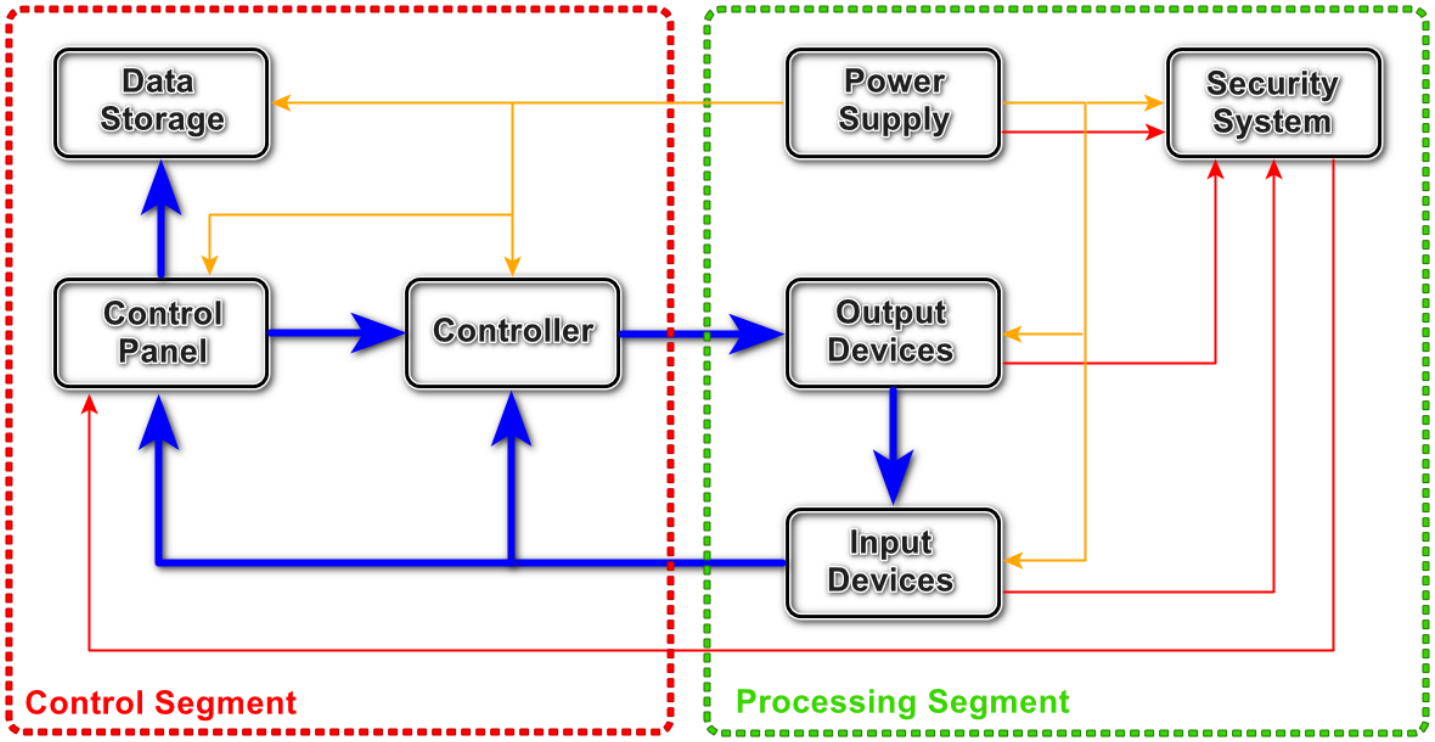


Figure 1.1 – Simple automated system diagram

4] Main Interface : they are the primary basic assembly of any automatic unit, at least for accomplishing a single cycle:

A] Input Devices: These devices gather information from the environment and send it to the control system like captures, sensors, switches and detectors and they can be classified by the signal they send to the controller.

Figure 1.2 – Input Devices with signal type

The original form of the signal of a input devices can be either digital which includes any capture uses 1 and 0 for communication, or analog which includes any physical measurement sensor so the signal is variable (example: -5v -> 0 -> 5v).

In some cases, the control unit can accept receiving variable (analog) signals, simply because they weren't designed for that purpose, having only digital pins, so in order to make it communicate with an analog capture, the signal must be transformed into digital using **ADC** integrated circuit (**A**nalog to **D**igital **C**onverter) which output the equivalent value of the input signal to a binary words or bytes but this result in the use of more pins.

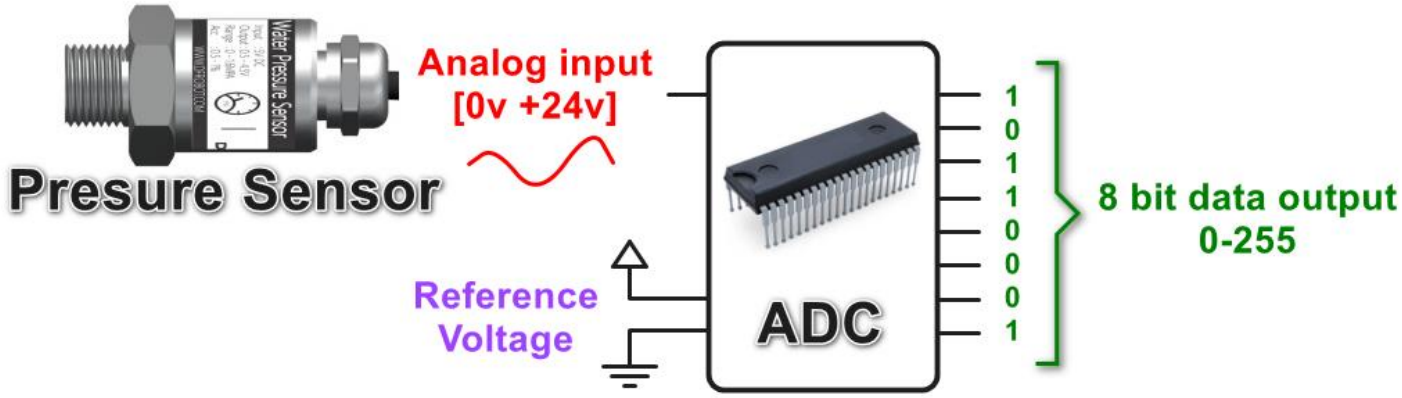


Figure 1.3 – Analog to digital converter

Note: if the ADC has more output pins for the output that means it has large resolution which offer better reading precision , for example: 8bit ADC represent 256 different levels while a 16bit ADC 65536 levels.

That creates a problem, a single sensor can occupy the whole controller pins for itself leaving us without any left pins to work with, also connecting all those wires to the control unit is inefficient and tedious, well there is a method to reduce the large amount of wires used by a single sensor to 2 wires only, by using another integrated circuit called the I2C which is a serial bus interface protocol circuit connected directly to the output of the ADC, and send the data to the controller.

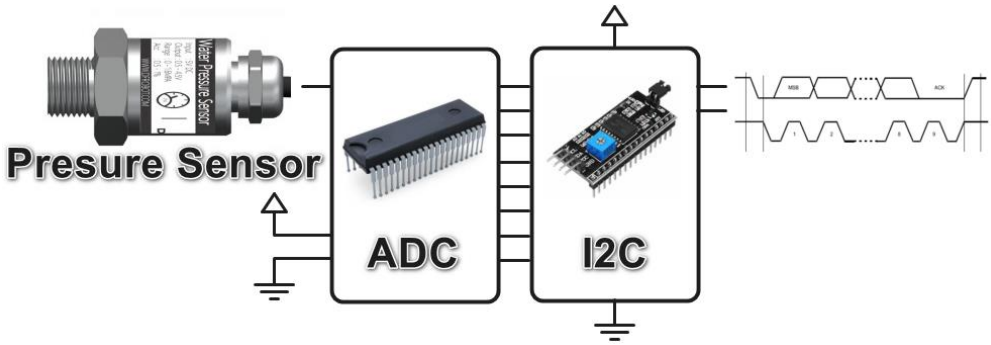


Figure 1.3 – Analog to digital converter equipped with I2C to reduce the number of wires

- The output pins of the I2C are SDA which represent the data and SCL is used to synchronize the data transfer between the master and slave devices on the bus.

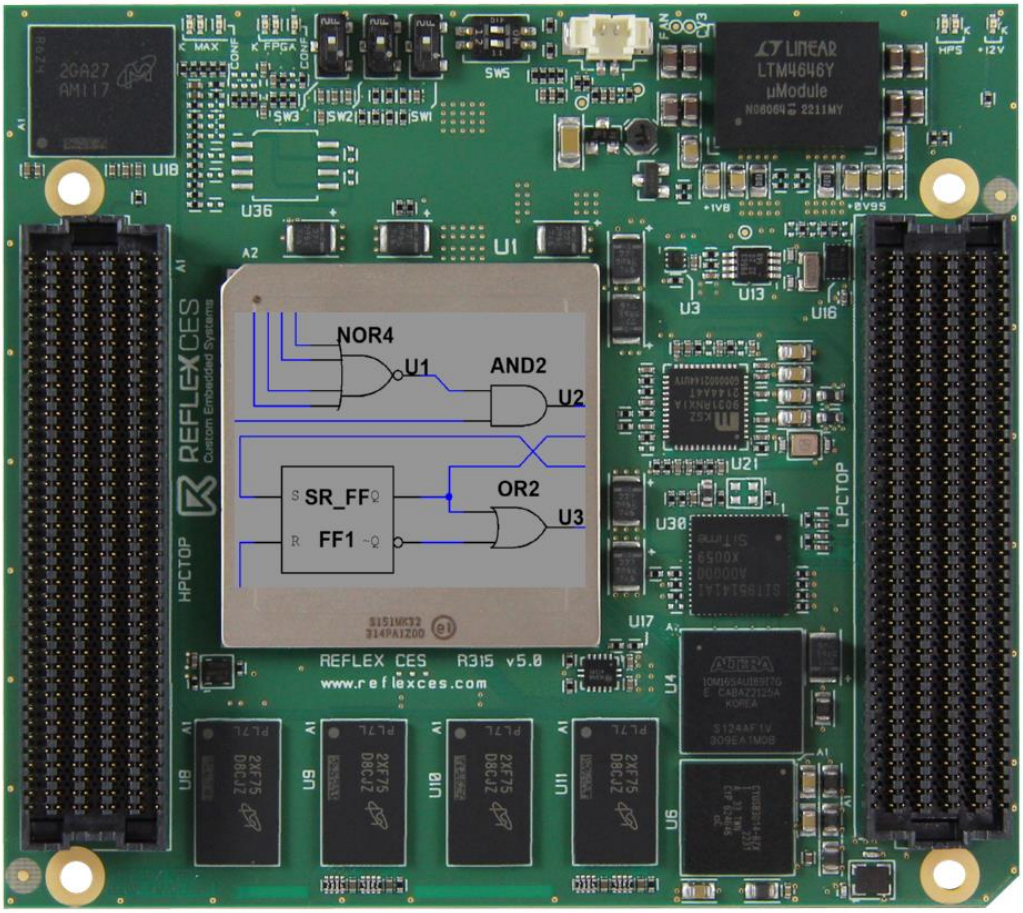
- Captures that uses serial protocol require less wires which make free pins for the controller to plug more devices.

B] Output Devices: Or “Actuators” and that’s what they do, by receiving a signal from an capture or a controller and translate it into physical actions, these devices actuate an object or a substance regardless of its state (Solid, Liquid or Gas) based on the coming signal, being an output device also means that it capable of output a digital information using a specific display.

Figure 1.4 – Output Devices (DC motor, Hydraulic Device)

In modern automation actuators can take various forms such as motors (DC, AC and stepper motors), hydraulic cylinders, solenoids, thermal changers…etc. And this type of hardware require a lot a power to operate if we consider them running in an industrial environment, so obviously we can’t connect them directly to a controller since the maximum voltage of its output is 24volt, These controllers are designed to handle low-power signals for control logic, data processing, and communication purposes. When it comes to actuators that require higher power levels, additional components such as **relays**, **motor drives**, or **power modules** are used to interface between the controller and the actuators.

FPGA


NOR4
U1
AND2
U2
SR_FF
FF1
OR2
U3
REFLEX CES
Custom Embedded Systems
HPCTOP
LPCTOP
REFLEX CES R315 v5.0
www.reflexces.com
LINEAR
LTM4646Y
µModule

1] Introduction

Choosing a controller to command an automated system depends on what the actual controller offers to the user, whether it is the quantity of pins, analog input/output support and modularity, but the most important thing is the operator's ability to deal with that controller like programming, reprogramming, connecting any compatible component, debugging, diagnostic and maintenance in case of a sudden malfunction....

2] Definition: What is an FPGA?

FPGA, short for Field-Programmable Gate Array, it's a unique type of controllers that offers a playground of possibilities for users to create and test custom logic circuit designs from scratch. Unlike traditional microcontrollers or processors, FPGAs don't come with a fixed set of circuits right out of the box. Instead, they're made of logic cells or logic array blocks which are **configurable units** to implement various digital functions. So it's like blank canvas waiting for your digital schematic.

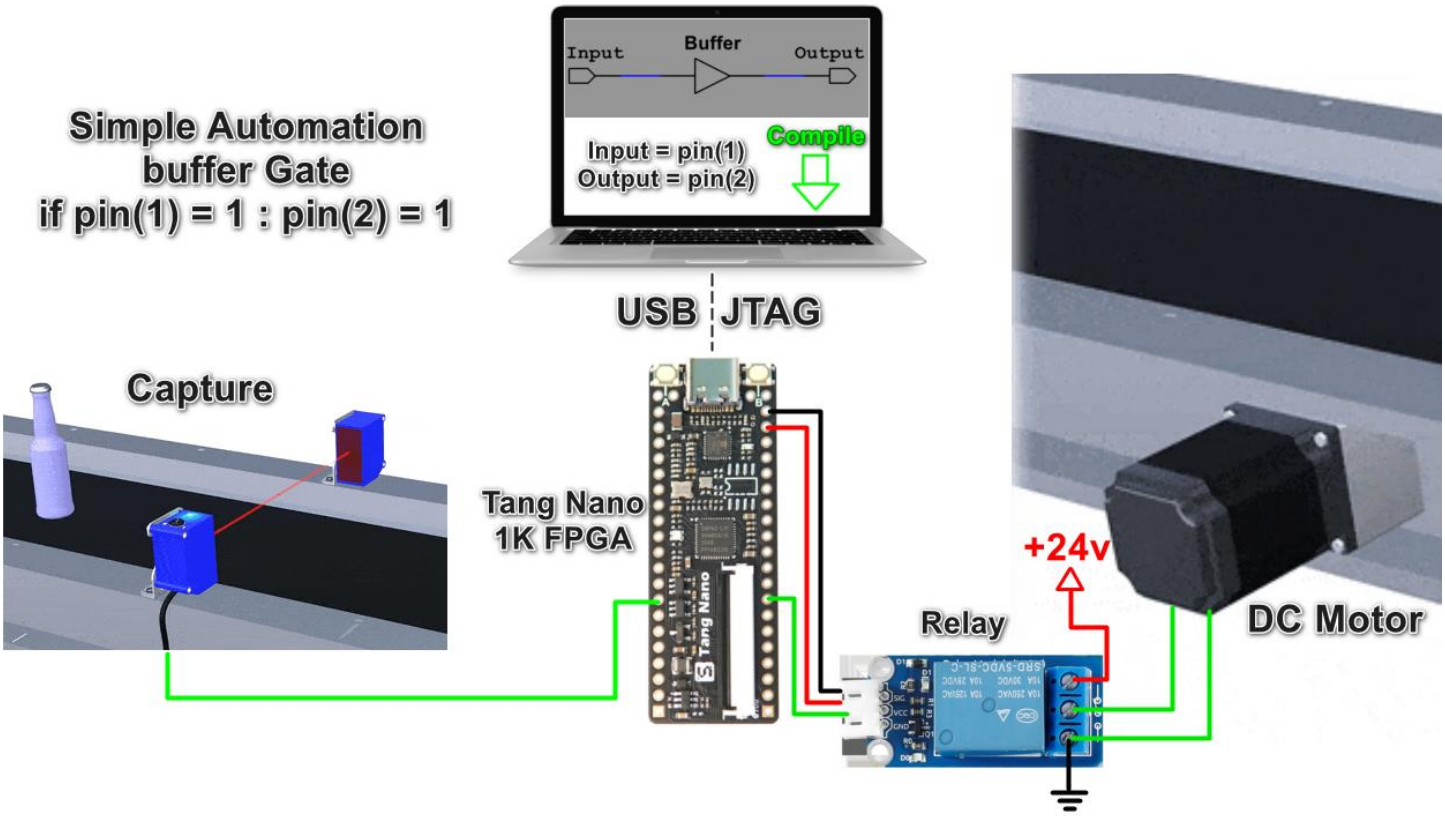


Figure 2.1 – Motor control using IR sensor and Buffer gate in a FPGA board

3] FPGA Board Components

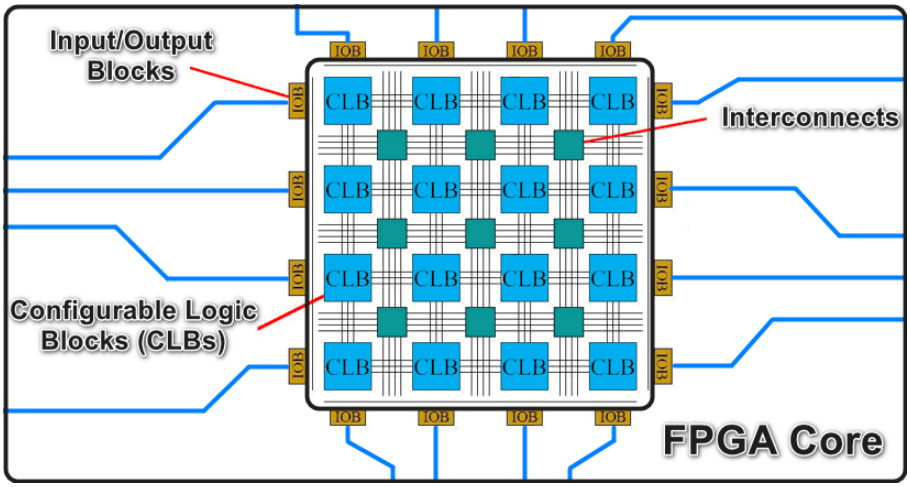


Figure 2.2 – inside the FPGA main chip

FPGAs are neither microcontrollers nor microprocessors; they are reconfigurable hardware devices that can be programmed to perform a wide range of tasks, including those of both microcontrollers and microprocessors, depending on the design and programming, but the FPGA chip itself cannot run properly without a board. An industry FPGA board is consist of:

3]1] FPGA Core chip

It's the main integrated circuit (SMD Soldered) that run all the operations on the PCB Board, and its vary depends on the manufacture, these chips are consist of three elements:

a] Configurable Logic Blocks (CLBs)

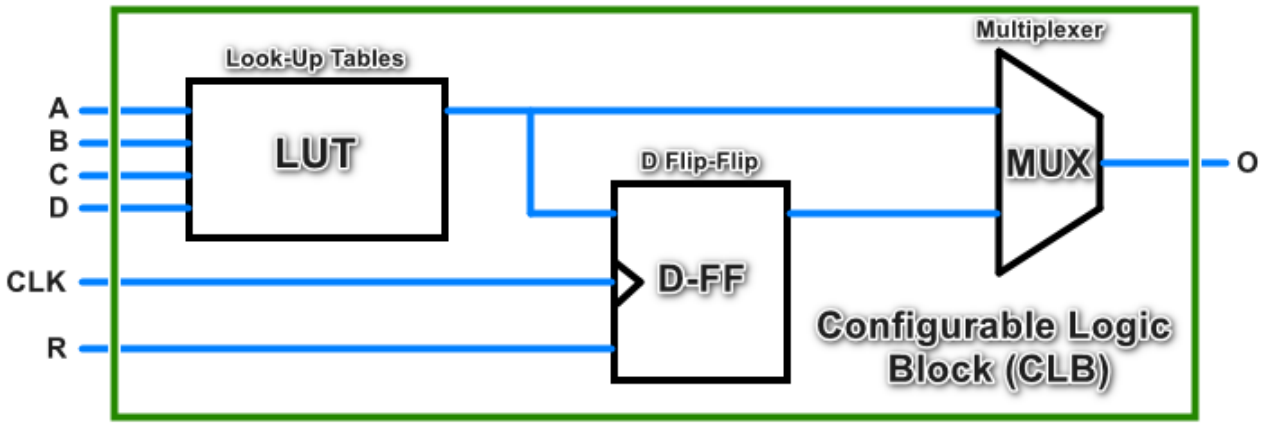


Figure 2.3 – inside on of the CLBs in the FPGA core

CLBs are logic circuits contains flip-flops (or registers), multiplexers and Look-Up Tables (LUTs), standing up to its name, CLBs are the components that make the FPGA a configurable controller by storing truth tables or small data tables to implement logic functions

b] Interconnects: FPGAs have a mesh of programmable interconnects that allow the CLBs to be connected to each other and to the input/output blocks (IOBs). These interconnects are typically implemented using a mix of programmable switches, multiplexers, and wiring resources.

c] Input/Output Blocks (IOBs): These blocks provide interfaces for connecting external signals to the FPGA. They include buffers that condition the signals going in and out of the FPGA, ensuring proper voltage level and signal integrity. And also I/O pads which are the physical pins on the FPGA package that connect to external devices.

3]2] JTAG Port (Joint Test Action Group Port)

The JTAG port is a standardized interface used for testing and debugging integrated circuits, including FPGAs. It allows for communication with the internal logic of the FPGA chip, enabling tasks such as programming, debugging, and boundary scan testing. For example, a JTAG port can be used with a USB-based JTAG programmer, which connects to a computer via USB and interfaces with the FPGA's JTAG pins. This allows engineers to upload FPGA configurations, perform real-time debugging, and conduct in-system testing of the FPGA design. The JTAG port provides a convenient and standardized method for hardware debugging and verification in FPGA development.

Some FPGAs doesn't have onboard JTAG integrated circuit so it requires to plug an external JTAG module to transfer the bit-stream code by USB.

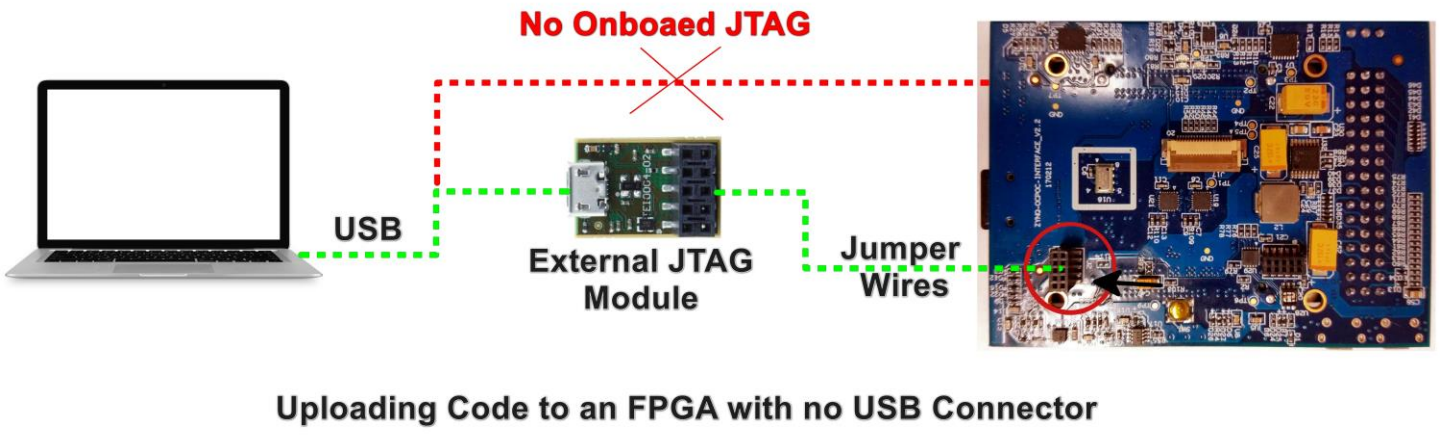


Figure 2.4 – Uploading with and without JTAG

3]3] User Flash Memory

It's the component that used to store the configuration bitstream code into the FPGA, this memory is non-volatile erasable memory, meaning it retains its data even when power is turned off, and start loading configuration to the FPGA core when the power is turned on, so it acts like a ROM in a computer motherboard.

3]4] Block SRAM

Block SRAM (Static Random-Access Memory) it is a fast volatile memory used for storing data during the operation of the FPGA., so it loses its contents when power is turned off, it is often used as on-chip memory for storing variables, buffers, and other data needed by the FPGA's logic, basically it acts like a RAM in a computer but it can be located inside the FPGA SoC (System on Chip) and separated from the Core.

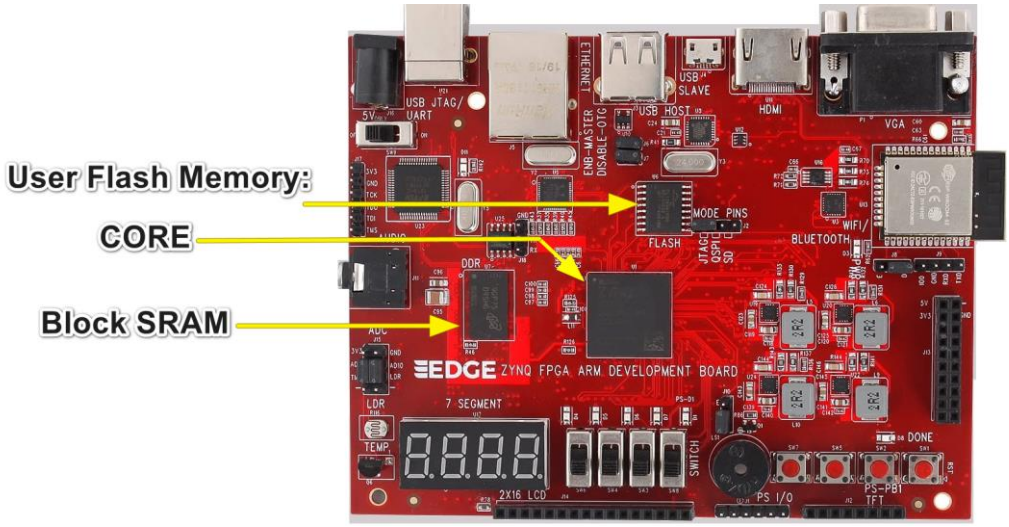


Figure 2.5 – FPGA Board (pointing at memory chips)

In Industrial FPGAs (typically the large class) the memories are usually located on the board (soldered) for easier diagnostic or replacement, on the other hand small footprint FPGAs can has both memories inside the SoC (System on Chip)

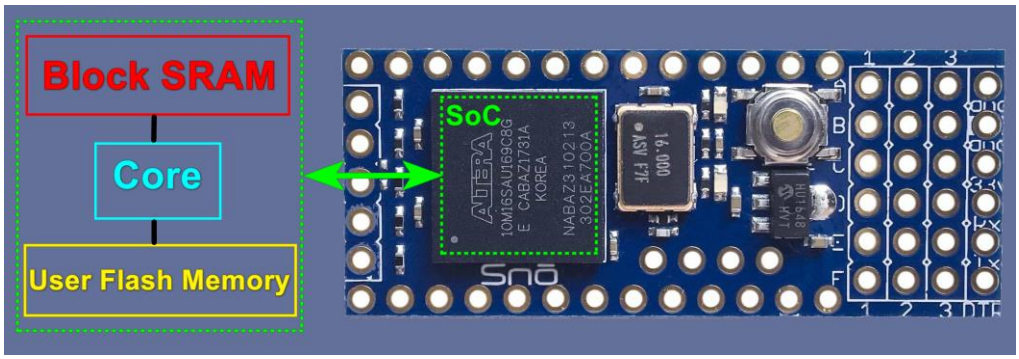


Figure 2.6 – FPGA Board (pointing at SoC)

3]4] Clock Management

The clock circuitry consists of:

- Clock generator: which is a crystal oscillator powered by the board to create high frequency signal (MHz).

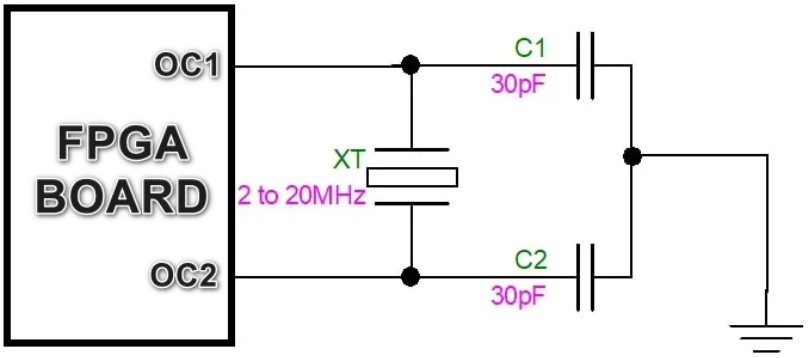


Figure 2.7 – crystal oscillator circuit

- Clock dividers: Using a simple binary counter to create slower clock signals for parts of the design that don't require high-speed operation or to make 1Hz signal for the RTC (**R**eal **T**ime **C**lock).

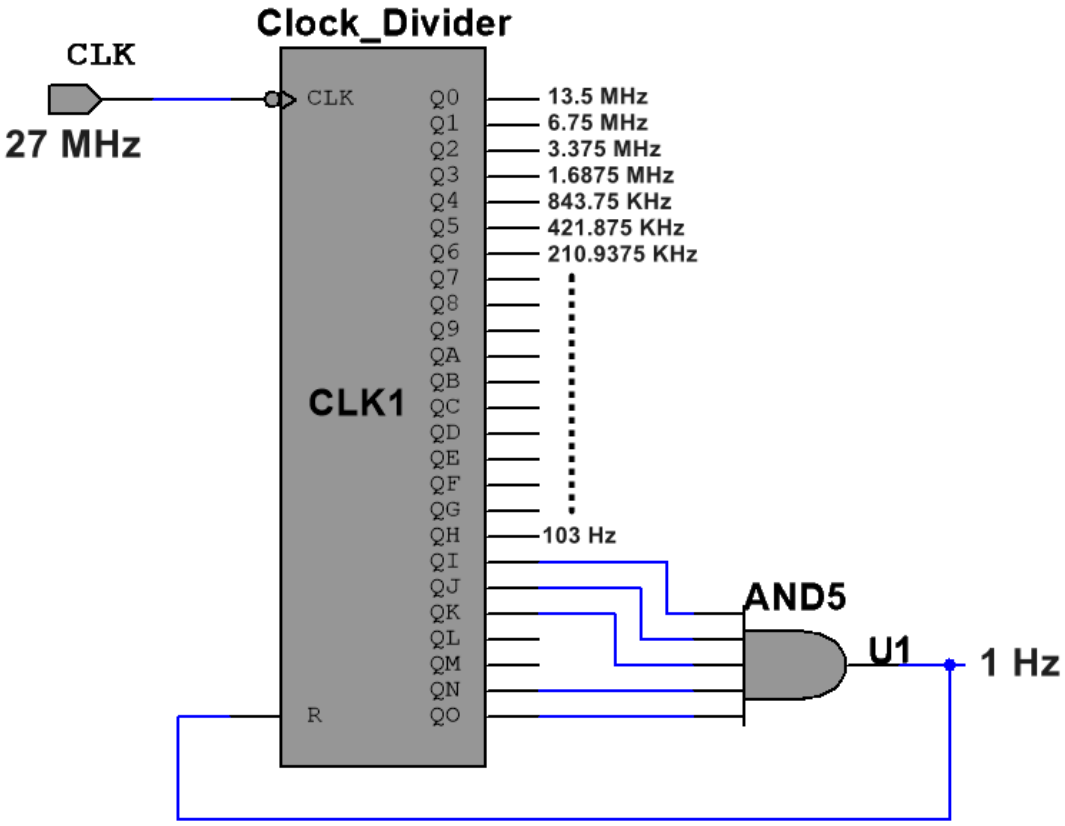


Figure 2.8 – Clock Divider

- Phase-Locked Loops (PLLs): PLLs are a key component of Clock Management in FPGAs. They are used to generate precise and stable clock signals with adjustable frequency and phase.

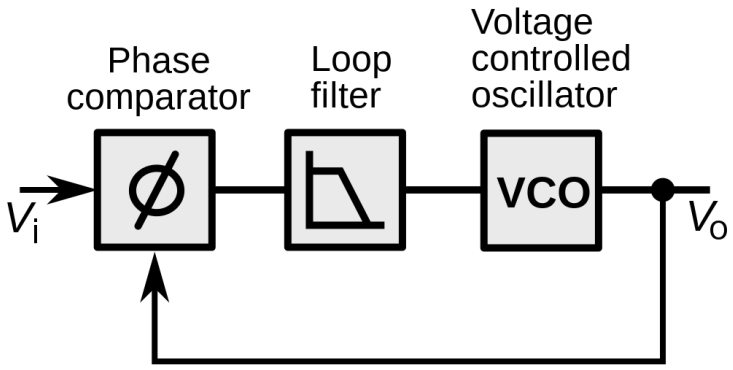


Figure 2.9 – Phase-Locked Loop

4] Configuration Methods: How to an FPGA?

There are generally 3 main methods for programming an FPGA (Field-Programmable Gate Array):

4]1] Hardware Description Languages (HDL)

HDLs are specialized programming languages used to describe the behavior of digital circuits. The two most common HDLs are **Verilog** and **VHDL**.

Engineers write code in these languages to define the functionality of the digital logic circuits they want to implement on the FPGA.

The HDL code is then compiled into a configuration file (bitstream) that can be loaded onto the FPGA. Here it is a simple example of AND gate circuit in Verilog:

```
//AND gate using Verilog
module AND2(A,B,Y);
input A,B;
output X;
assign X = A & B;

endmodule
```

Advantages:

- **Low-Level Control:** HDLs like Verilog and VHDL provide fine-grained control over the design, allowing precise manipulation of hardware components.
- **Widely Used:** HDLs are industry-standard languages for FPGA design, with extensive community support, resources, and established best practices.
- **Suitable for Complex Designs:** HDLs are well-suited for complex designs with intricate timing requirements and detailed logic operations.
- **Hardware Optimization:** Designers have direct control over the hardware architecture, enabling optimization for performance, area, and power consumption.

Disadvantages:

- **Steep Learning Curve:** HDLs require a solid understanding of digital design concepts and syntax, making them challenging for beginners.
- **Verbose Syntax:** HDL code can be verbose and complex, leading to longer development cycles and potentially more error-prone designs.
- **Limited Abstraction:** HDLs operate at a low level of abstraction, which can make it cumbersome to express high-level algorithms or behavioral descriptions.
- **Time-Consuming:** Writing and debugging HDL code can be time-consuming, especially for large designs or when debugging complex timing issues.

4]2] High-Level Synthesis (HLS)

HLS tools allow engineers to write algorithms in high-level languages such as C, C++, or SystemC.

The HLS tool then automatically converts this high-level code into RTL (Register Transfer Level) code, which can be synthesized into an FPGA design.

This method abstracts the low-level details of FPGA programming, making it more accessible to software developers. Here it is a simple example of AND gate circuit in C++ :

```
#include "ap_int.h"
void and_gate(bool a, bool b, bool &y) {
    #pragma HLS INTERFACE ap_ctrl_none port=return
    #pragma HLS INTERFACE ap_none port=a
    #pragma HLS INTERFACE ap_none port=b
    #pragma HLS INTERFACE ap_none port=y

    y = a && b;
}
```

Advantages:

- **Abstraction:** HLS allows designers to describe hardware functionality at a higher level of abstraction using familiar languages like C, C++, or SystemC.
- **Productivity:** HLS can significantly reduce development time by enabling faster design iterations and eliminating the need for manual RTL coding.
- **Optimization:** HLS tools automatically optimize high-level code for performance, area, and power consumption, freeing designers from low-level optimization tasks.
- **Algorithm Exploration:** HLS facilitates algorithm exploration and hardware/software co-design by enabling rapid prototyping and experimentation.

Disadvantages:

- **Limited Control:** HLS abstracts away low-level hardware details, which can limit the designer's control over the final implementation.
- **Tool Maturity:** HLS tools may not yet support all language features or optimizations compared to traditional HDL-based flows, leading to potential limitations or suboptimal results.
- **Verification Complexity:** Verifying HLS-generated RTL code may be challenging due to the complexity of optimizations and transformations performed by the tool.

4]2] Schematic Entry

Engineers use a graphical interface to draw the desired circuit using raw integrated circuits, logic gates, wires and connectors to function like a real circuit but inside the FPGA core, The tool then generates the corresponding HDL code or bitstream that configures the User Flash Memory. Some FPGAs main software supports schematic-to-code function but the best and most common used third-party software to generate code from a logic gates circuit is **Multisim** from **National Instrument**, this software is compatible with all FPGAs IDE and support both Digital and PWM signals but lacks the analog I/O processing.

Here it is a simple example of AND gate using schematic entry by Multisim software and FPGA Gowin IDE:

1- Drawing the circuit: AND gate with 2 inputs (A,B) and 1 output (X)

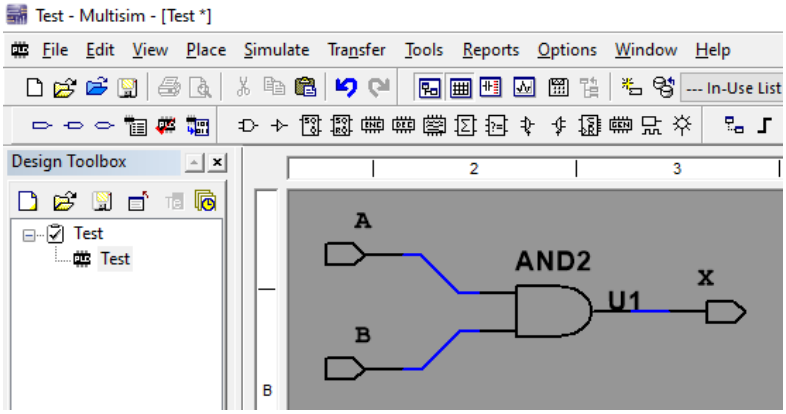


Figure 2.10 – Multisim window (Building Schematic Entry Project)

2- Generating the equivalent code: by clicking on the “Export to PLD”, Multisim will generate 2 files, 1st file (top level module) contains the connectors and wiring across all the circuit, and the other file (Package) contains the function library of every logic gate or integrated circuit that have been drawn in the software.

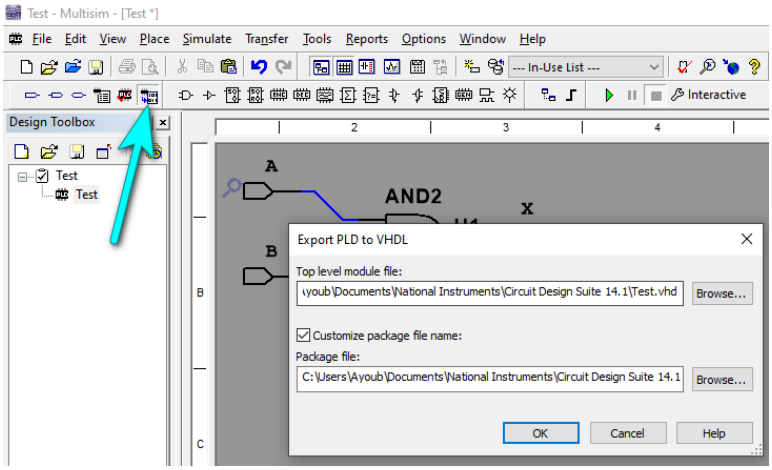


Figure 2.11 – Multisim window (Exporting PLD files)

3- Loading the 2 files into the FPGA IDA: we use the Gowin IDE that came with the Gowin 1K Tang Nano FPGA, after creating a new project and select the board we load the 2 generated files.

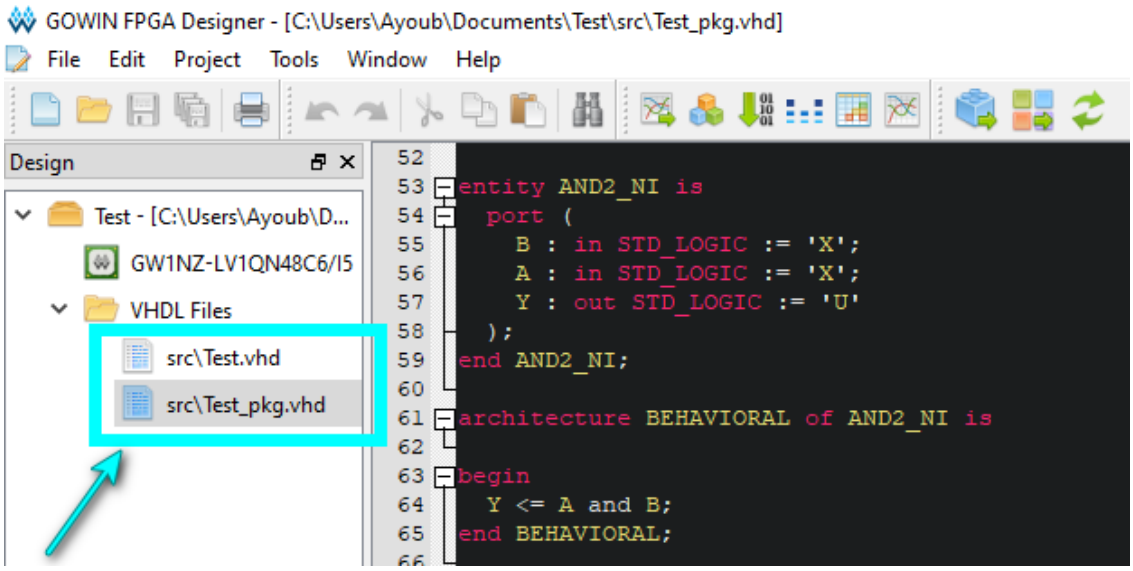


Figure 2.12 – Gowin IDE window (Uploading the PLD files)

4- Creating the “Physical Constraints File”: the file is necessary because it connects the circuit that we draw to the physical pins of the actual FPGA board by declaring the connectors to the corresponded pin number on the board (Optional).

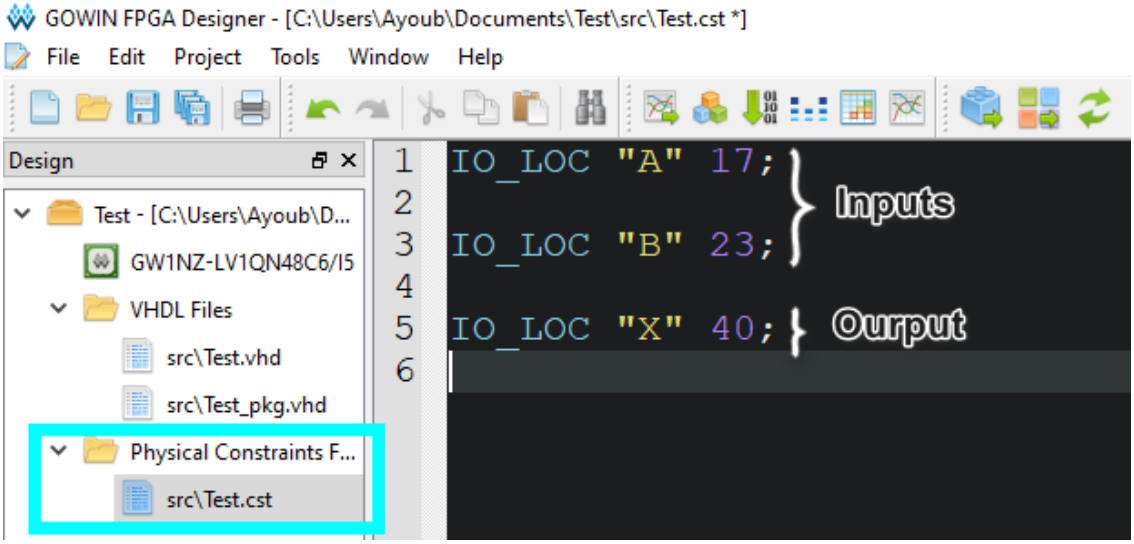


Figure 2.13 – Gowin IDE window (Creating Pins Connections)

5- Connecting the project peripherals: for a simple test of the “AND gate” we connect 2 buttons [A to pin (17) and B to pin (23)] and a visual output like a LED to see the output of the circuit.

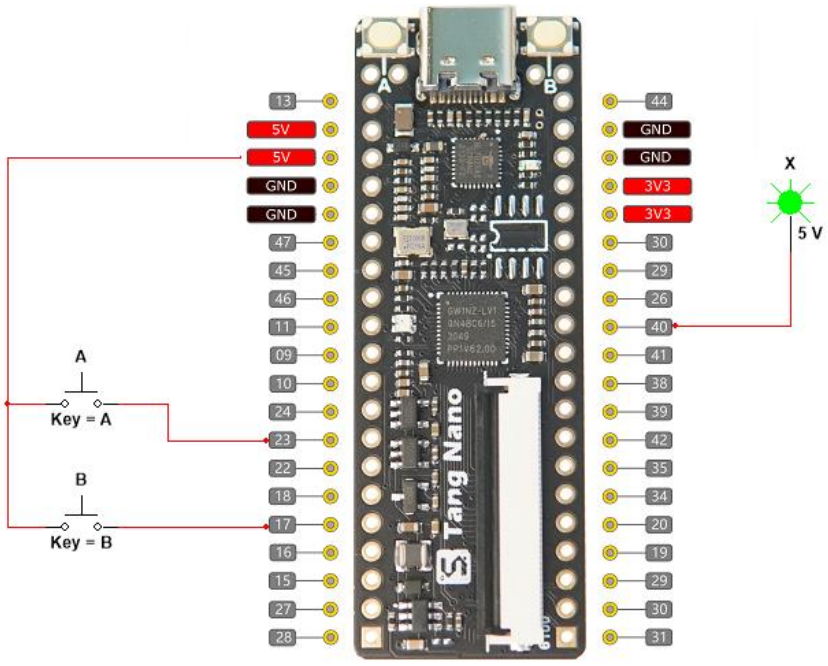


Figure 2.14 – Project Representation using Gowin Tang Nano 1K

6- Compiling and programing: we connect the FPGA to the computer by the USB C connector with onboard JTAG which make it a lot easier, then we compile the code to Bitstream file and then flash it to the FPGA by the USB cable.

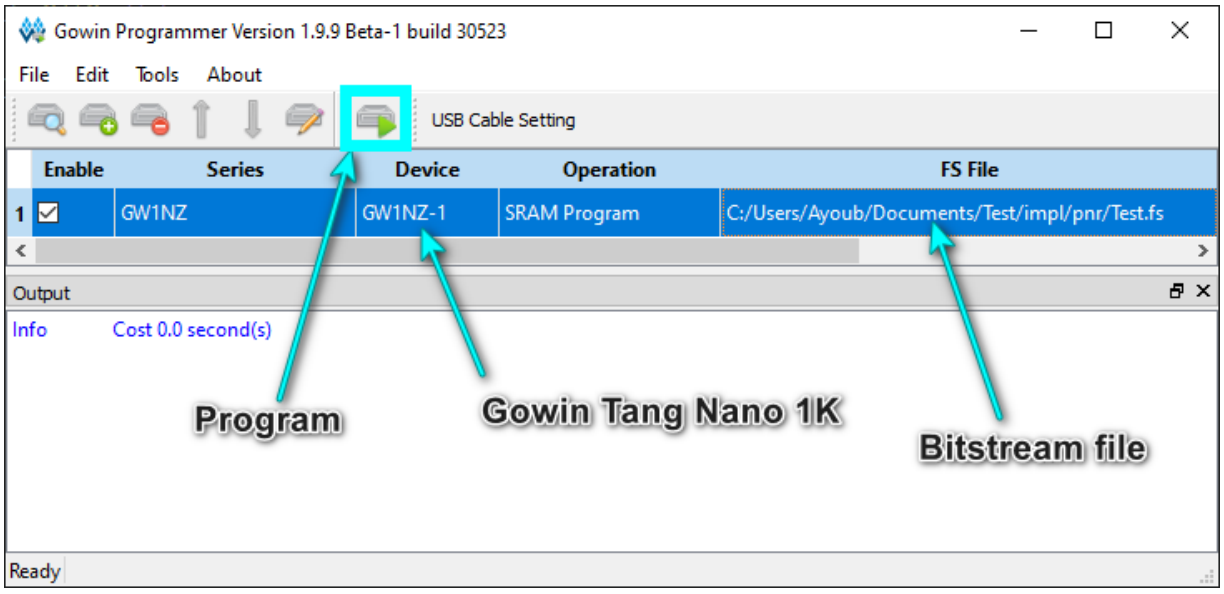


Figure 2.15 – Gowin Programmer window (Uploading Bitstream file to the FPGA)

We can also select the storage type for the program from “Operation”, so by selecting the SRAM memory, the code will stay as long as the power is on, and by selecting the Flash memory, the code will be stored even when the power is off.

Advantages:

- **Intuitive Design:** Schematic entry provides an intuitive graphical representation of the circuit, making it accessible to designers with limited programming experience.
- **Visual Debugging:** Visual representations facilitate debugging and visualization of the design's behavior, aiding in error identification and correction.
- **Rapid Prototyping:** Schematic entry enables quick prototyping of simple designs without the need for writing code, making it ideal for beginners or for exploring design concepts.

Disadvantages:

- **Limited Scalability:** Schematic entry may become cumbersome and less efficient for large and complex designs due to the manual nature of the process.
- **Resource Constraints:** Libraries may not always include all necessary components, requiring custom components for certain functionalities, which can limit design flexibility.
- **Version Control:** Managing revisions and collaborative development can be challenging compared to text-based HDL code, which supports version control systems.

5] Uses and Benefits of using FPGA

Digital Signal Processing (DSP):

FPGAs are widely used in DSP applications such as audio and video processing, speech recognition, and image processing.

They offer high-performance parallel processing capabilities, making them suitable for real-time signal processing tasks.

Communications and Networking:

FPGAs are used in networking equipment such as routers, switches, and network interface cards (NICs) for packet processing, protocol parsing, and encryption/decryption.

They provide flexible and programmable solutions for implementing various networking protocols and algorithms.

Embedded Systems:

FPGAs are integrated into embedded systems for tasks such as motor control, robotics, industrial automation, and automotive electronics.

They offer high-speed I/O interfaces, customizable peripheral interfaces, and real-time processing capabilities, making them suitable for demanding embedded applications.

High-Performance Computing (HPC):

FPGAs are increasingly being used in HPC applications such as high-frequency trading, computational finance, and scientific computing.

They provide parallel processing capabilities and can be reconfigured on-the-fly to adapt to changing algorithms or workload requirements.

Artificial Intelligence (AI) and Machine Learning (ML):

FPGAs are used to accelerate AI and ML algorithms in applications such as neural network inference and training.

They offer high-performance compute resources, low-latency processing, and energy efficiency, making them suitable for AI/ML workloads.

Security and Cryptography:

FPGAs are used in security-sensitive applications such as encryption/decryption, secure communication protocols, and hardware security modules (HSMs).

They provide hardware-level security features, customizable cryptographic algorithms, and tamper-resistant designs.

Benefits of FPGAs:

Flexibility and Reconfigurability:

FPGAs can be reconfigured and updated with new functionality or algorithms after deployment, offering flexibility for evolving requirements and future-proofing designs.

They allow for rapid prototyping, iteration, and customization of hardware designs without the need for costly and time-consuming ASIC development.

Performance and Parallelism:

FPGAs offer parallel processing capabilities with multiple logic elements operating simultaneously, resulting in high-performance computing for parallelizable tasks.

They can achieve low-latency and high-throughput processing, particularly for tasks that benefit from parallelism such as signal processing and data-intensive computations.

Customization and Optimization:

FPGAs enable designers to customize hardware architectures to meet specific application requirements, optimizing performance, power consumption, and resource utilization.

They allow for hardware acceleration of critical tasks, improving overall system performance compared to software-based approaches running on microcontrollers (MCUs) or general-purpose processors.

Low Power Consumption:

FPGAs offer power-efficient solutions for compute-intensive tasks, with the ability to selectively activate or deactivate logic elements based on workload demands.

They provide energy-efficient processing for battery-powered or power-constrained applications, such as LoT devices and portable electronics.

Time-to-Market Advantage:

FPGAs can accelerate time-to-market for new product development by enabling rapid prototyping, iteration, and verification of hardware designs.

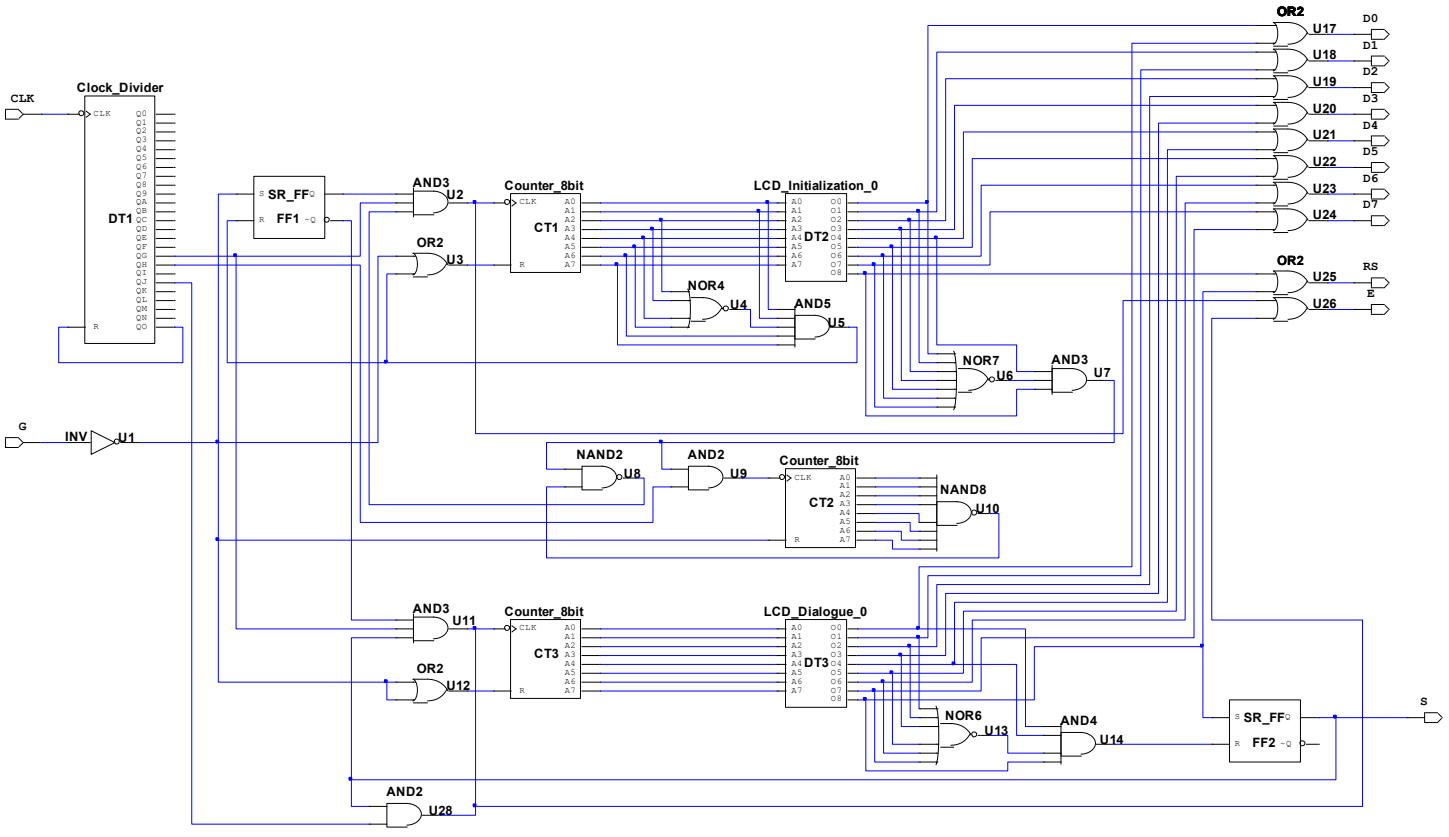
They allow for concurrent hardware and software development, reducing development cycles and enabling faster product iterations.

Scalability and Integration:

FPGAs can be used alone or integrated with other processing elements (e.g., CPUs, GPUs) within a single chip or system-on-chip (SoC) architecture.

They offer scalability in terms of computational resources, I/O interfaces, and peripheral integration, supporting diverse application requirements and system configurations.

Binary logic
Schematic design



1] Introduction

As it known in physics, there are 2 sates of power, On and Off. That wasn't much of a deal back in mid-19th century when large, bulky and heavy circuit board dominated the market where ICs including microcontrollers and ASICs didn't exist yet. So it was the only option despite the inconvenient circuit assembly and the components form factor. Even though the Boolean algebra was discovered in 1847, but scientist weren't know that is the key to develop a new technology beyond basic electric circuits.

Figure 3.1 – The first form of transistors (1947)

The transistor was invented in December 1947 at Bell Laboratories by physicists John Bardeen, Walter Brattain, and William Shockley. This invention revolutionized electronics and paved the way for modern technology by replacing bulky and power-hungry **vacuum tubes** with smaller, more efficient solid-state components.

This invention changed the world, not only it replaced physical switches for high power lines, it also was the main component of **logic gates** which function based on the Boolean algebra.

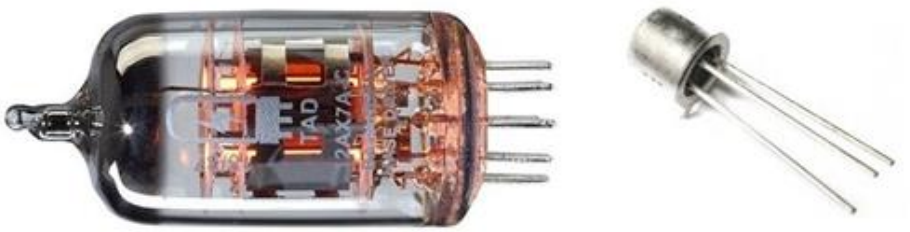


Figure 3.2 – Old transistor vs New Transistor

BJT Transistors (Bipolar Junction Transistor) act like a switch, it have three pins which are the **Base**, the **Collector** and the **Emitter**, so the goal here is to control the current flow that passes from the collector to the emitter by providing or cutting power of the base pin:

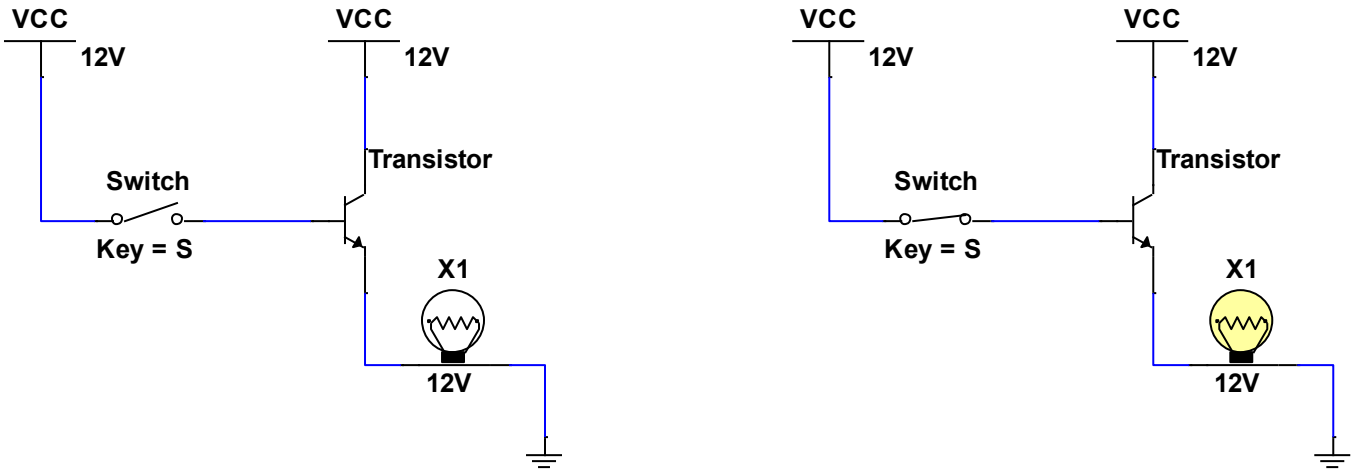


Figure 3.3 – Current Flow Control Using Transistor and Switches

2] Binary System

Concurrently, researchers were exploring Boolean logic, a mathematical system developed by George Boole in the mid-19th century. Boolean logic deals with binary variables and operations such as AND, OR, and NOT.

Engineers realized that they could combine transistors and Boolean logic to create electronic circuits that performed logical operations. By designing circuits that implemented Boolean functions based on **1** and **0** according to base-2 numbering system.

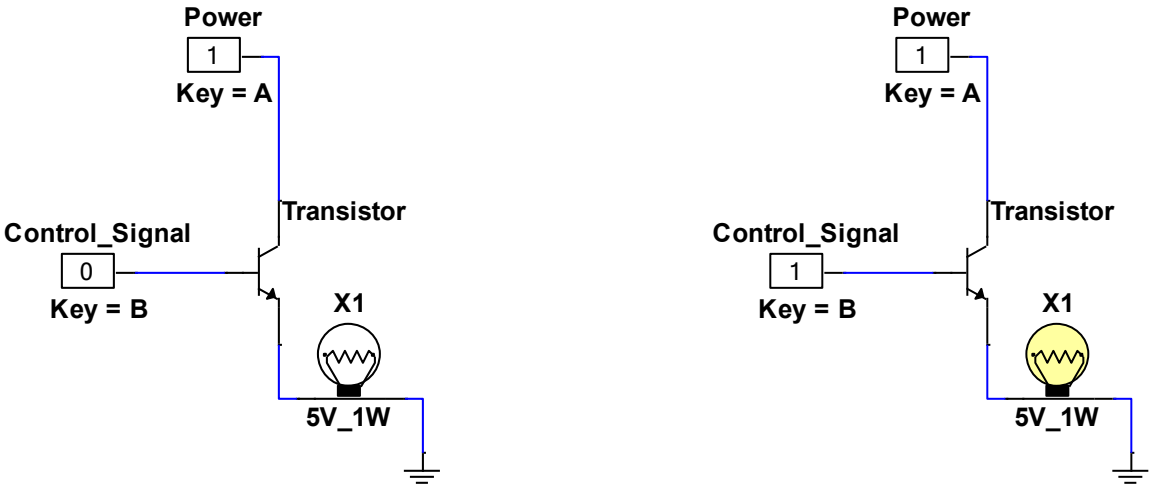


Figure 3.4 – Current Flow Control Using Transistor and Control Signals

3] Logic Gates

3]1] AND Gate

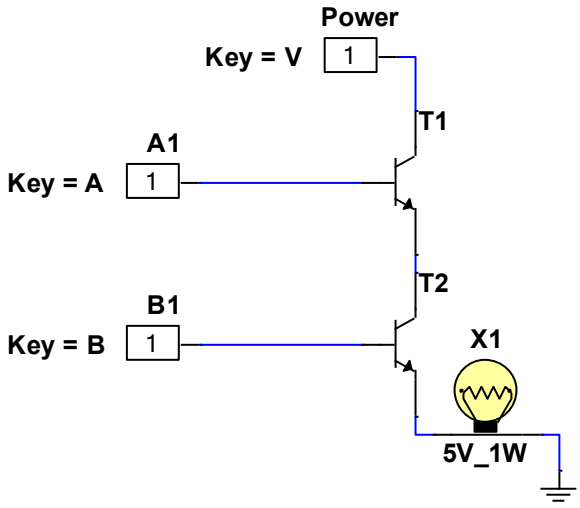
The AND gate strictly requires all the input signals to be “1” in order to make the power passes to the charge, because the transistors are mounted in series so it represent the AND operation (*) in Boolean algebra or the ∧ (logical AND) symbol.

0 * 0 = 0

1 * 0 = 0

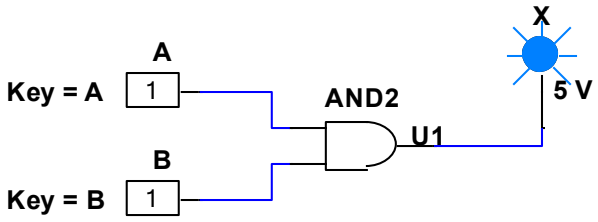
1 * 1 = 0

1 * 1 = **1**



Truth table and Symbolic representation :

A	B	**X**
0	0	0
1	0	0
0	1	0
1	1	**1**



AND Gate Integrated Circuits: **TTL (74LS08)**, **CMOS (CD4081)…**

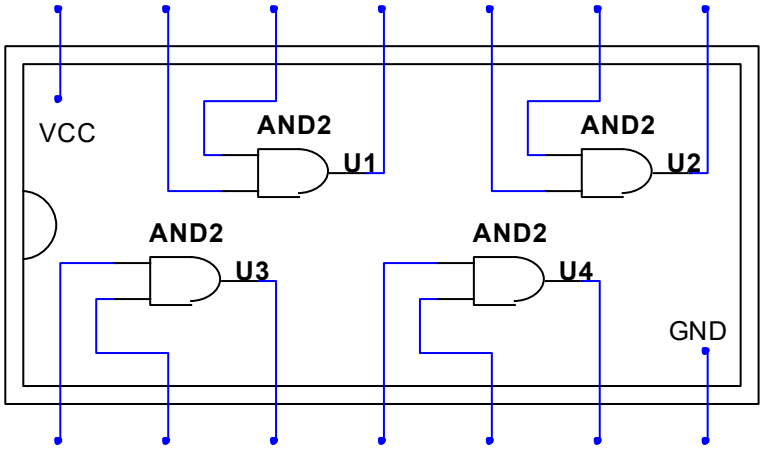


Figure 3.5 – AND Gate Integrated Circuit

3]2] OR Gate

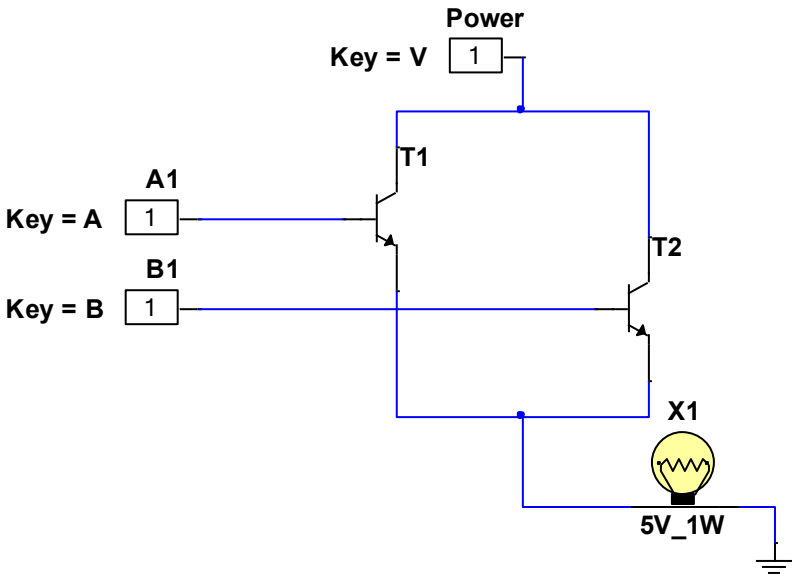
The Or gate requires **atleast** one input signal to be “1” in order to make the power passes to the charge, because the transistors are mounted in parallel, and it represent the OR operation (+) in Boolean algebra or the **V** (logical OR) symbol.

0 + 0 = 0

1 + 0 = **1**

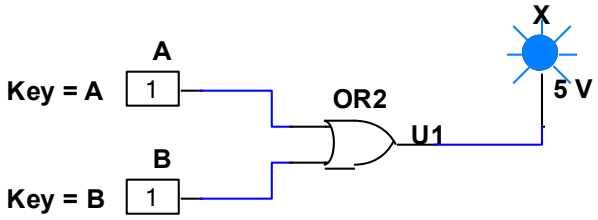
0 + 1 = **1**

1 + 1 = **1**



Truth table and Symbolic representation :

A	B	X
0	0	0
1	0	**1**
0	1	**1**
1	1	**1**



OR Gate Integrated Circuits: **TTL (74LS32)**, **CMOS (CD4071)…**

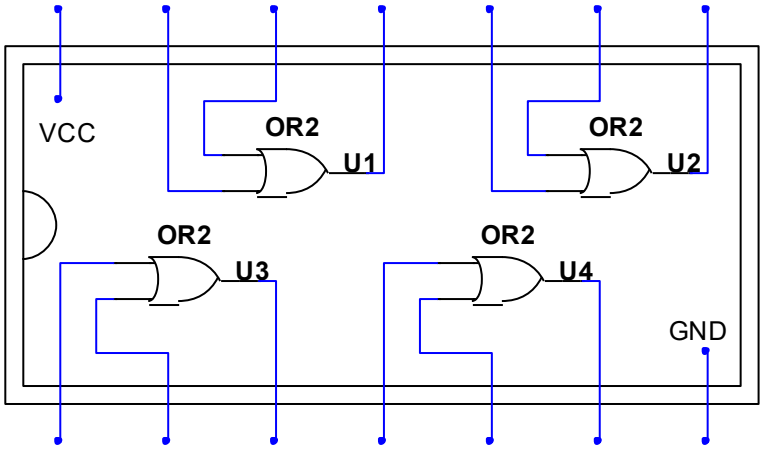


Figure 3.6 – OR Gate Integrated Circuit

3]3] NOT Gate

The NOT gate **reverse** the input signal by connecting the output before the transitor and directly to the power (VCC) , so when there is no input signal ”0”, the output is “1” since its connected to VCC , and when the input signal is “1”, then the output is 0 because the transistor make the power pass to the ground (GND).

Reversed values are written with BAR “-” on top of the value letter

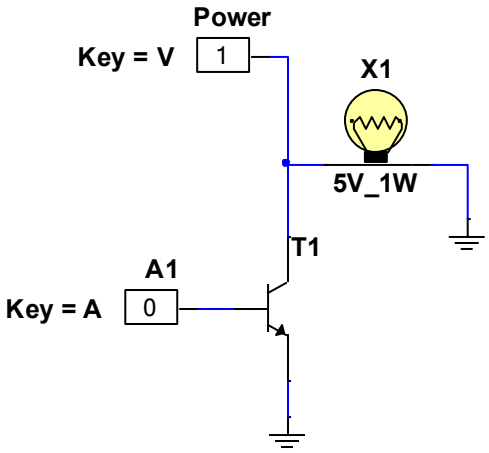
0 = **1**

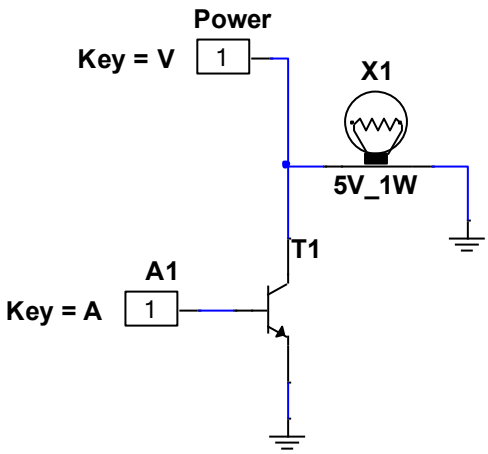
= 0

$\overline{A} = X$

$A = \overline{X}$

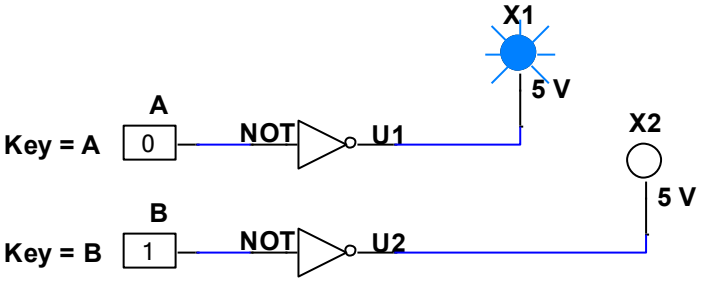
1





Truth table and Symbolic representation :

A	**X**
1	0
0	**1**



NOT Gate Integrated Circuits: **TTL (74LS04)**, **CMOS (CD4069)…**

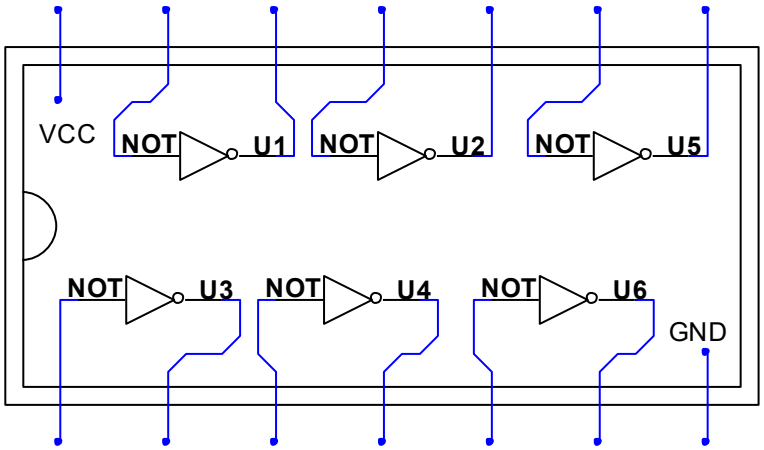


Figure 3.7 – NOT Gate Integrated Circuit

3]4] XOR Gate

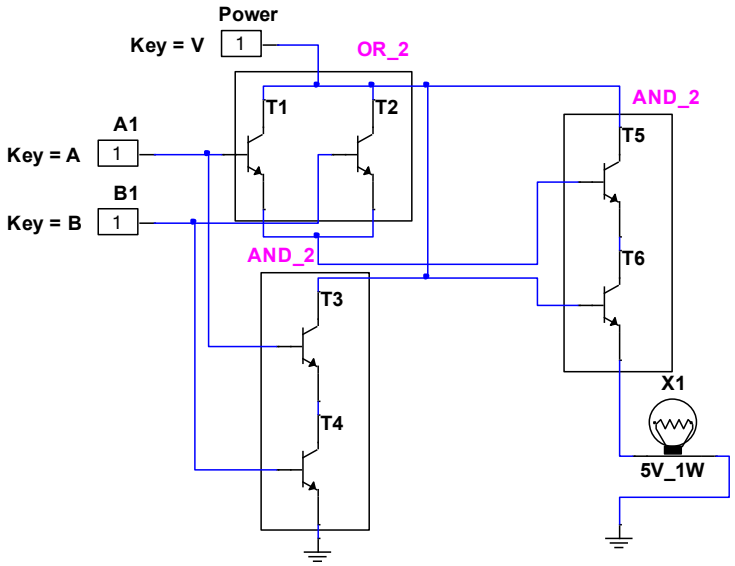
The XOR gate (Exclusively-OR) it a combination of 3 logic gates (AND,OR and NOT) , if one of the input siganls is “1” then the output is 1, but when all the input signals are “1” then the signal is “0”. The boolean formula for an XOR can be expressed using OR gate AND a AND gate + NOT gate : (A + B)*($\overline{A * B}$) or A⊕B

0 + 0 = 0

1 + 0 = **1**

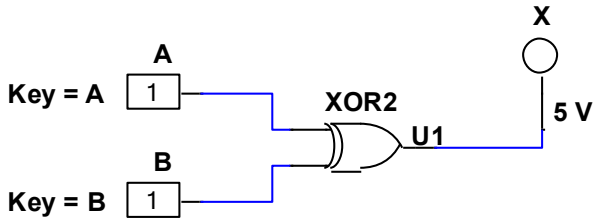
0 + 1 = **1**

1 + 1 = 0



Truth table and Symbolic representation :

A	B	**X**
0	0	0
1	0	**1**
0	1	**1**
1	1	0



OR Gate Integrated Circuits: **TTL (74LS32)**, **CMOS (CD4071)…**

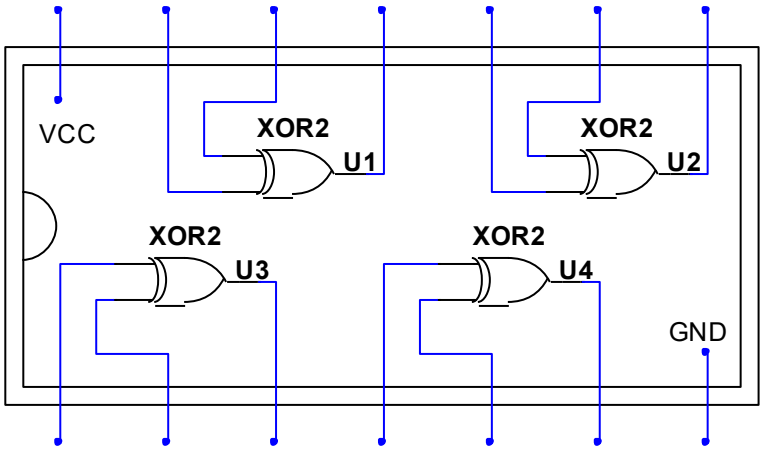


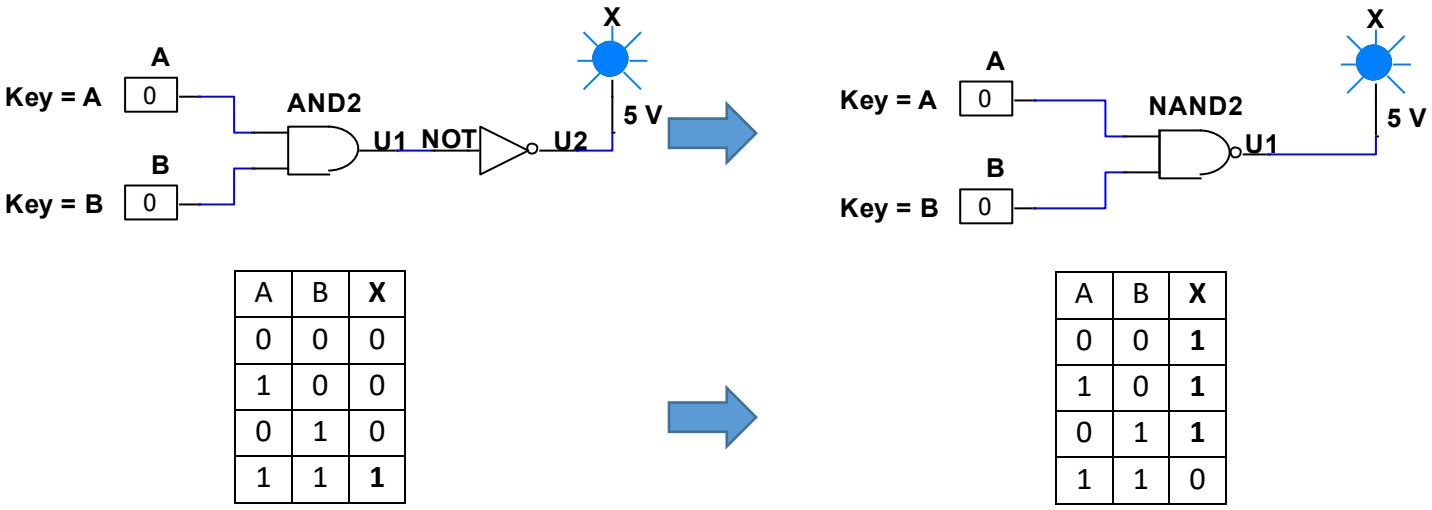
Figure 3.8 – XOR Gate Integrated Circuit

3]5] Inverted Logic Gates

NOT Gate can be added at the output of any logic gates to invert it signal, inverted logic gates used to **compensate** or **reduce** the number of logic gates in the circuit.

To simplify the design, we logic gates with small circle at the end of the shape to declare an inverted logic gate.

AND Gate:

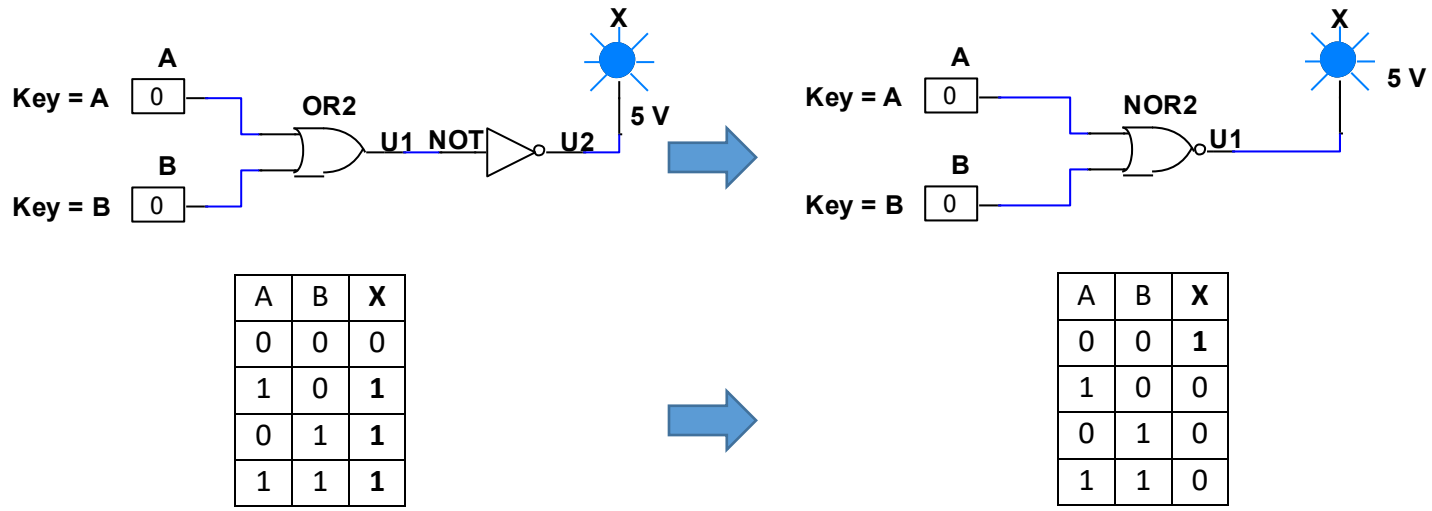


A	B	X
0	0	0
1	0	0
0	1	0
1	1	**1**

A	B	X
0	0	**1**
1	0	**1**
0	1	**1**
1	1	0

Figure 3.9 – Inverted AND Gate

OR Gate:

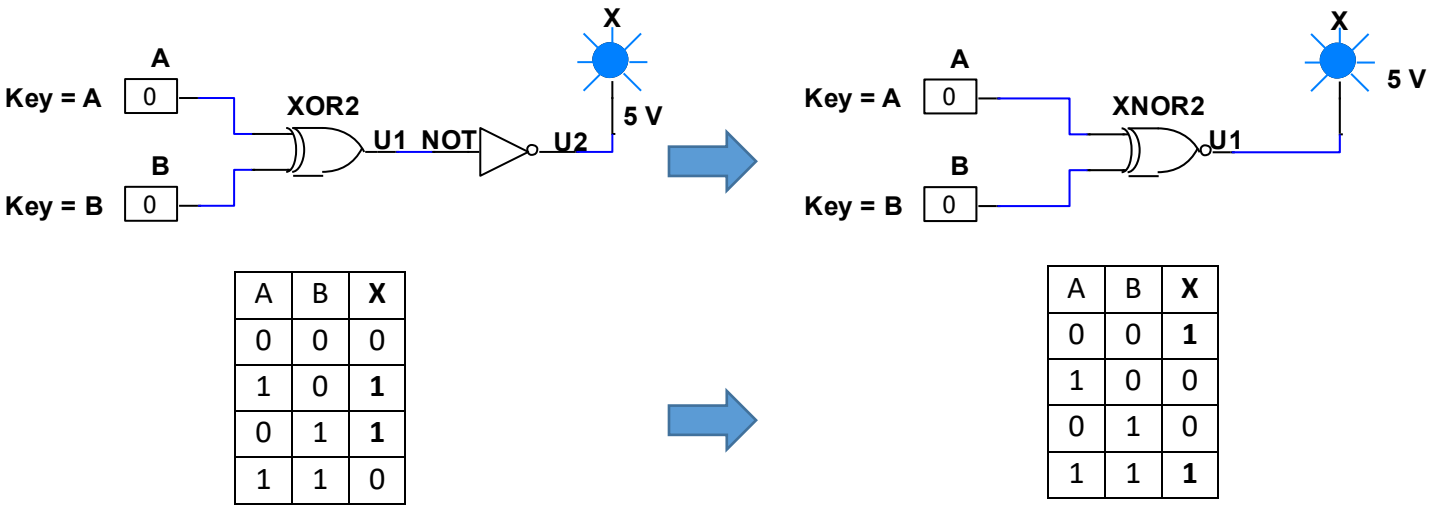


A	B	X
0	0	0
1	0	**1**
0	1	**1**
1	1	**1**

A	B	X
0	0	**1**
1	0	0
0	1	0
1	1	0

Figure 3.10 – Inverted OR Gate

XOR Gate:

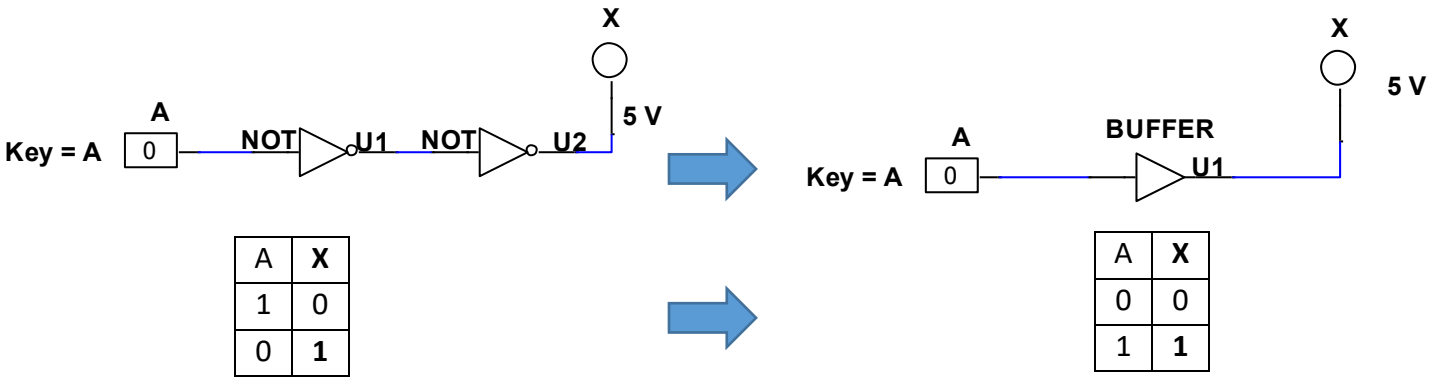


A	B	X
0	0	0
1	0	**1**
0	1	**1**
1	1	0

A	B	X
0	0	**1**
1	0	0
0	1	0
1	1	**1**

Figure 3.11 – Inverted XOR Gate

NOT Gate: Adding a NOT Gate to another Not Gate cancel the inversion effect getting a logic gate called a **Buffer Gate**:



A	X
1	0
0	**1**

A	X
0	0
1	**1**

Figure 3.12 – Inverted NOR Gate (Buffer)

Buffers are considered as logic gates, but they basically act like a **wire**, so the input signal equal to the output signal, but buffers are important for integrated circuits to purify and filter the incoming signal to make it as digital as possible for the chip.

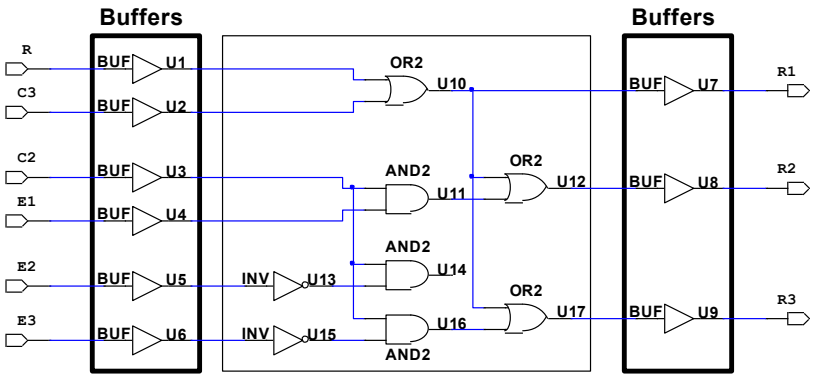


Figure 3.13 – Logic Circuit surrounded by Buffers

Note: All Logic Gates (including Inverted gates) can have up to 8 inputs in common commercial ICs, the complexity of the IC increases significantly, making it less practical and cost-effective. For applications requiring more inputs, designers often use multiple smaller gates in combination or opt for more specialized ICs such as multiplexers, decoders, or programmable logic devices (PLDs), which can handle larger numbers of inputs and offer more flexibility.

4] Flip Flops

4]1] Definition

A flip-flop is a digital circuit element capable of storing binary data in the form of a binary state (typically represented as 0 or 1). It has two stable states and is capable of changing its output state in response to a clock signal or control input. Flip-flops are used to store and synchronize data in digital systems.

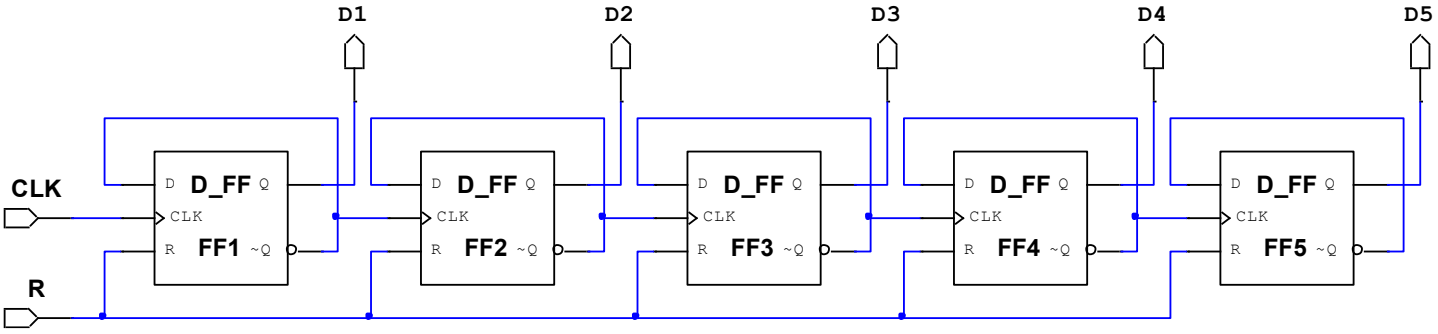
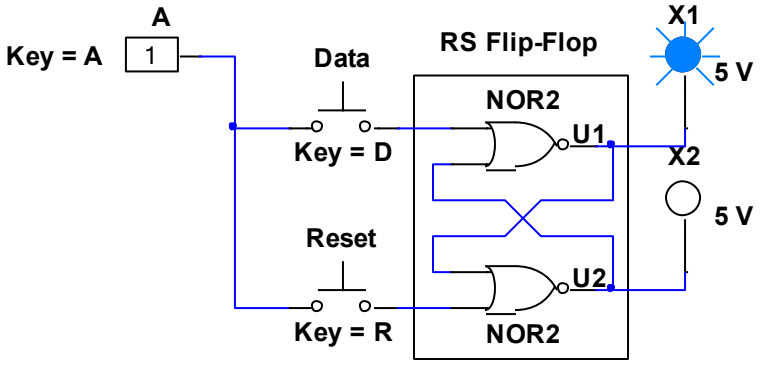


Figure 3.14 – Register using D Flip Flops

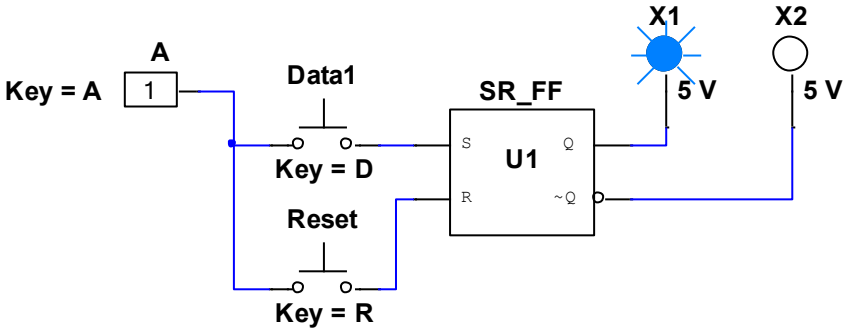
Logic gates would never maintain their output data state, but if we connect 2 NOR logic gates to each other, we will create a glitch loop that makes the signal stay in one of the NOR gate as long as we don't change the signal in the second NOR gate. Then the signal will jump to the next triggered gate keeping the signal in a closed loop even if there is no input signal "1".

4]3] SR Flip-Flop

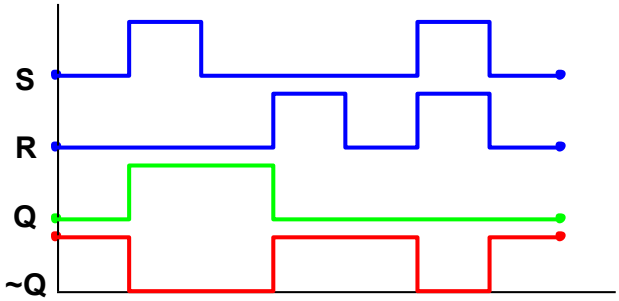
SR Flip-Flop is simply a **basic Flip-Flop**, constructed of 2 NOR Gates latched to each other to make a signal loop. By storing one data signal (1bit), that makes it the smallest unit data storage. SR Flip-Flop consists of 2 input pins (S for data and R for reset) and 2 output pins (Q for output and ~Q which is the inverted Q output).



Symbolic representation:



Timing Diagram:



Uses: SR Flip-Flops used for signal toggle, or making memory modules by stacking 4, 6 or 8 SR Flip-Flops.

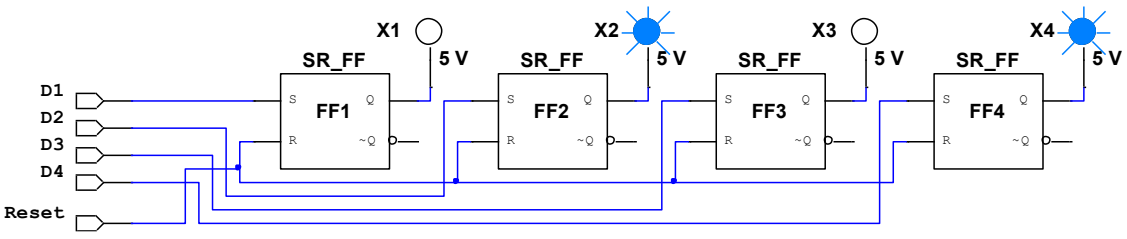
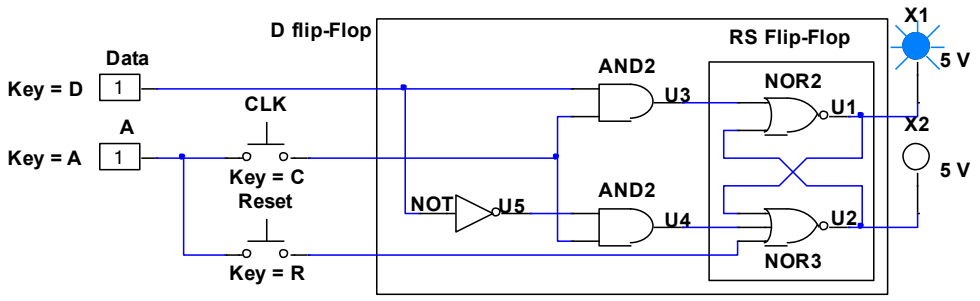


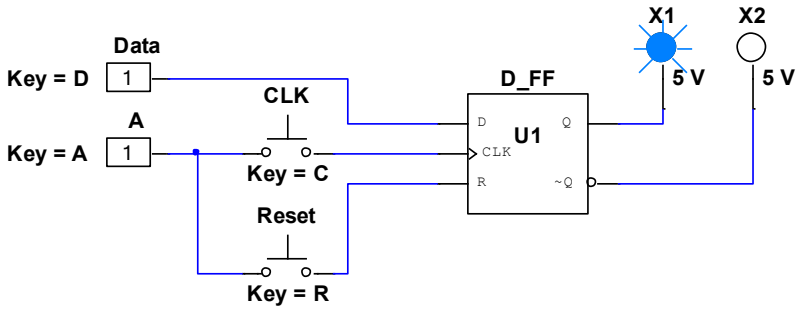
Figure 3.15 – 4bit Data Storage using SR Flip-Flops

4]3] D Flip-Flop

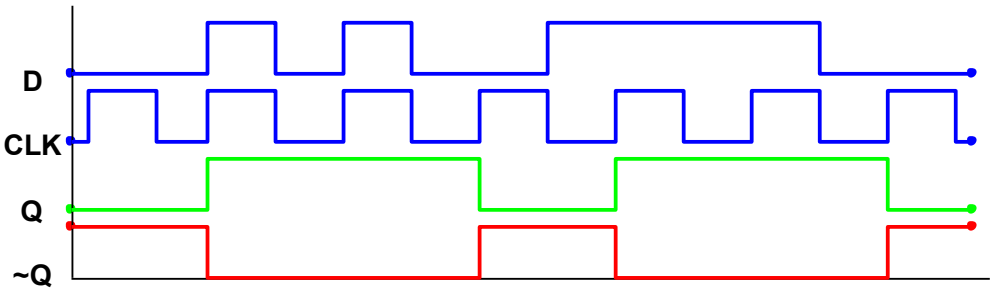
D Flip-Flip is Synchronous Basic Flip-Flop, it allows to store or remove data specific clock edges, it consist of 3 input pins (D for data, CLK for clock and R for reset) and 2 output pins (Q for output and ~Q "the inverted Q").



Symbolic representation:



Timing Diagram:



Uses: D Flip-Flops used for Registers, signal displacement and simple counters

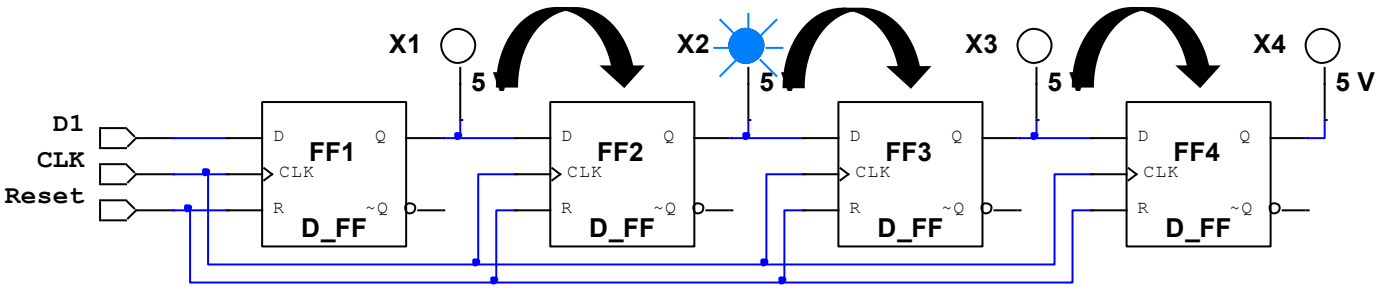
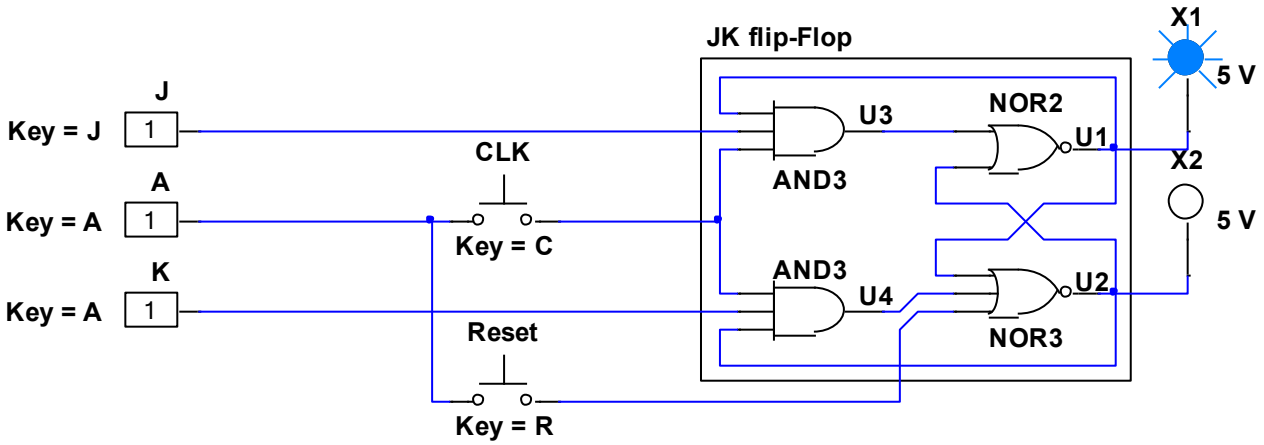


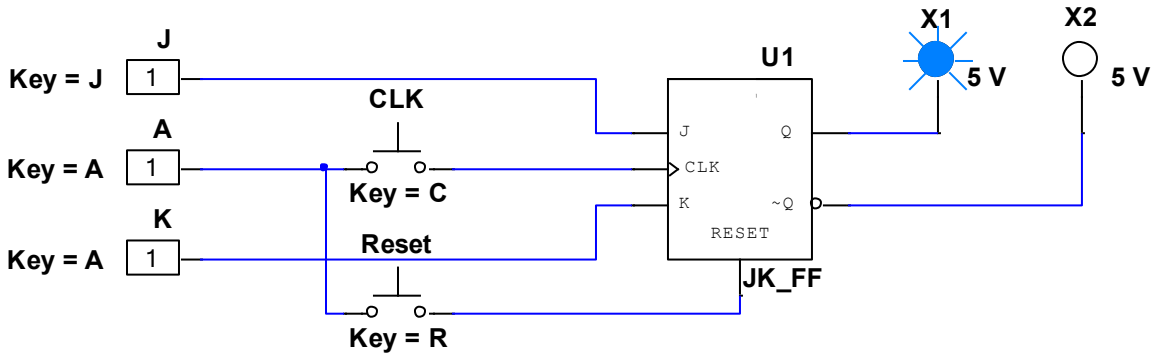
Figure 3.16 – 4 Steps Register Using D Flip Flops

4]3] JK Flip-Flop

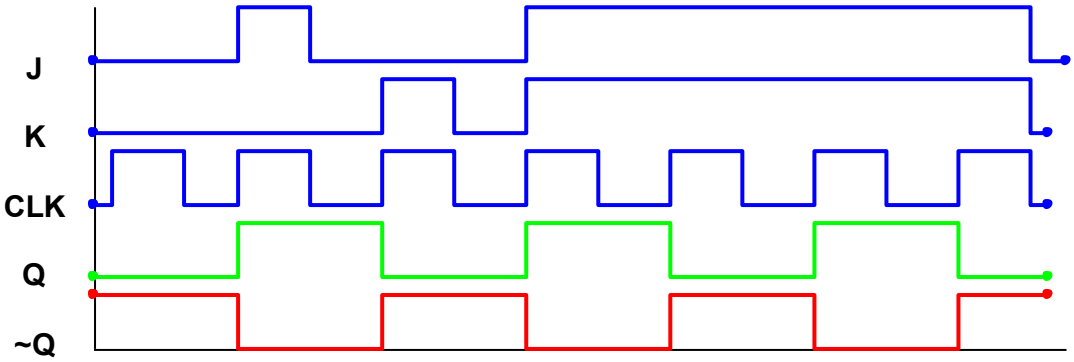
The JK Flip-Flop is a type of sequential logic circuit that can store one bit of data and it's an extension of the SR Flip-Flop with additional functionality to prevent the invalid state. It consists of clock input pin, J K input pins and Q and ~Q output pins.



Symbolic representation:



Timing Diagram:



Uses: JK Flip-Flops are versatile and can be configured in various ways to achieve specific circuits by changing JK signal, and they can be used to build counters with high accuracy by alternating toggled “Q” output signal between 1 and 0 each clock edge in condition of both JK input signals are “1”.

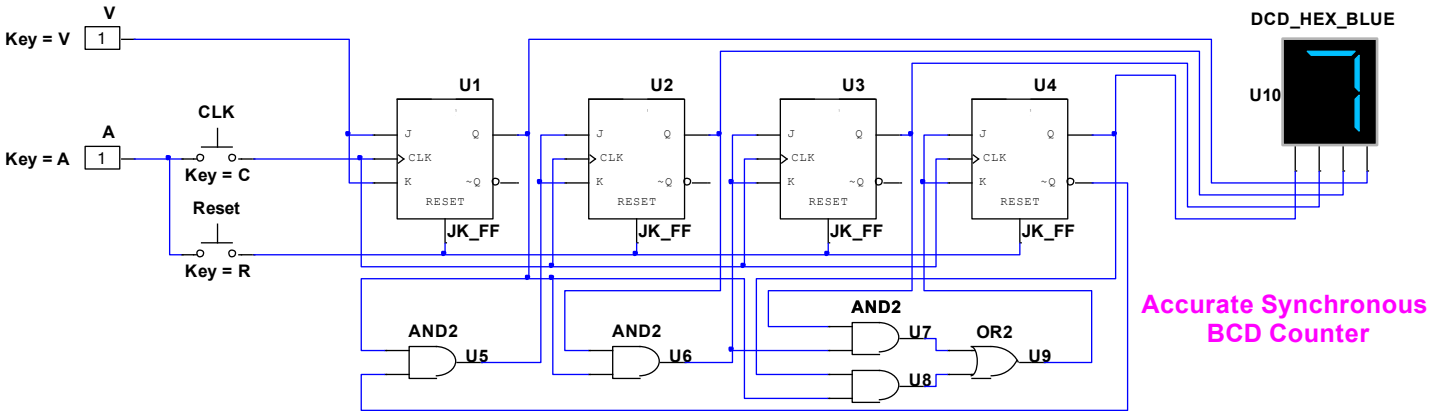


Figure 3.17 – 4bit Counter Using JK Flip-Flops

5] Schematic Entry Implementation

Programming languages, such as C++, Verilog, etc..., are used to write instructions or algorithms that computers can understand and execute. These languages involve specifying sequences of commands or operations to perform tasks, such as data processing, control flow, and logic operations. On the other hand, Schematic entry involves drawing logic circuits to design a project, is not typically considered a programming language. Instead, it's a graphical method used in electronic design automation (EDA) tools to create visual representations of electronic circuits.

Choosing the method that suits you really depends on your perspective to it and how you understand its fundamentals, creating your own project from scratch without copy-pasting proves your full knowledge to your chosen method regarding how hard it is, there will be some benefits and compromises for your choices so it’s important to know the field you work in, accepting it requirements and dealing with the uses drawbacks as a developer.

I chose schematic entry due it simplicity, ease and flexibility makes me to create large quantity of projects without learning any programming language at all, just by building actual working virtual binary circuits with graphical symbols such as logic gates or Flip-Flops and linking them together with wires and IO connecters allows me to fully understand the project or modify it at any given time, so when it comes to maintenance and debugging, schematic entry will make it simple for you.

Creating a FPGA Project Using Schematic Entry

5]1] Selecting the FPGA

There are plenty of FPGA models on the market now days, from tiny SMD chips for custom project, to small compact dev-board for tryout and development, to large industrial robust FPGA modules that withstand the harsh industry environment and power demand.

In Algeria, Choices are very limited (unless you order yourself). "**Spartan"** chip is the main core for some boards like Mimas, Edge and Elbert starting at a price point of 12,000.00 DZD for the base model and 48,000.00 for the high end model which include high core speed and large RAM capacity with wide user interface modules. **Artix-7 Nexys 4** is the most advanced and powerful FPGA on sell here in Algeria with a price tag of 70,000.00 DZD including DDR2 RAM, ADCs, PWM ports and a handful of user interface and captures.

For beginners, these prices is not a motivation to try a FPGA, even if they have money to buy one they will not use it to its full potential. Luckily there is one single option to obtain which is the **GOWIN TangNano 1K** from **GOWIN Semiconductor** manufacture with compact design, decent core speed and enough memory to store and operate so it will do the job, with a price of only 2900.00 DZD making it a cheaper board to start from, unfortunately the boards doesn't have ADCs meaning that it doesn't support Analog IO.

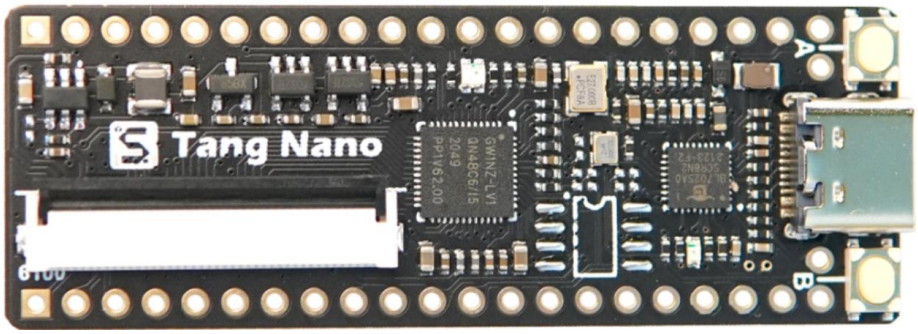


Figure 3.18 – Top View of The Gowin Tang Nano 1K

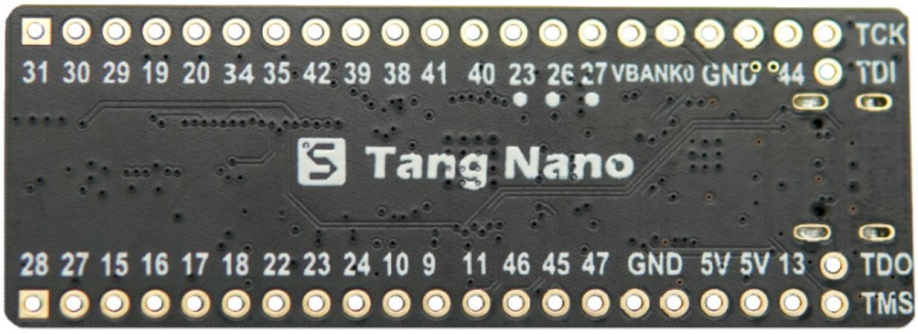


Figure 3.19 – Bottom View of The Gowin Tang Nano 1K

So we bought one brand new and this is the board/core specifications:

- Logic Module Units: **1152 Lut4**.
- Block SRAM: **72 Kb** (Kilobits).
- Flash Memory: **64 Kb** (Kilobits).
- Crystal Oscillator Frequency: **27 MHz**
- Core voltage: **1.2V**
- Phase-Locked Loop (PLLs): **1**
- Configuration Port: USB Type-C
- Debugger: Onboard BL702 chip provides JTAG.
- User Interface: 2 Buttons + 1 RGB LED + 40 pin screen display interface.
- Power Pins : 2 pins of **3.3V** + 2 pins of **5V** + 4 GND pins
- IO Pins: **32 Digital pins** support 4mA, 8mA, 16mA, 24mA with in dependent pull-up / pull-down resistor.
- Dimensions: 58.34mm x 21.29mm x 1.33mm.

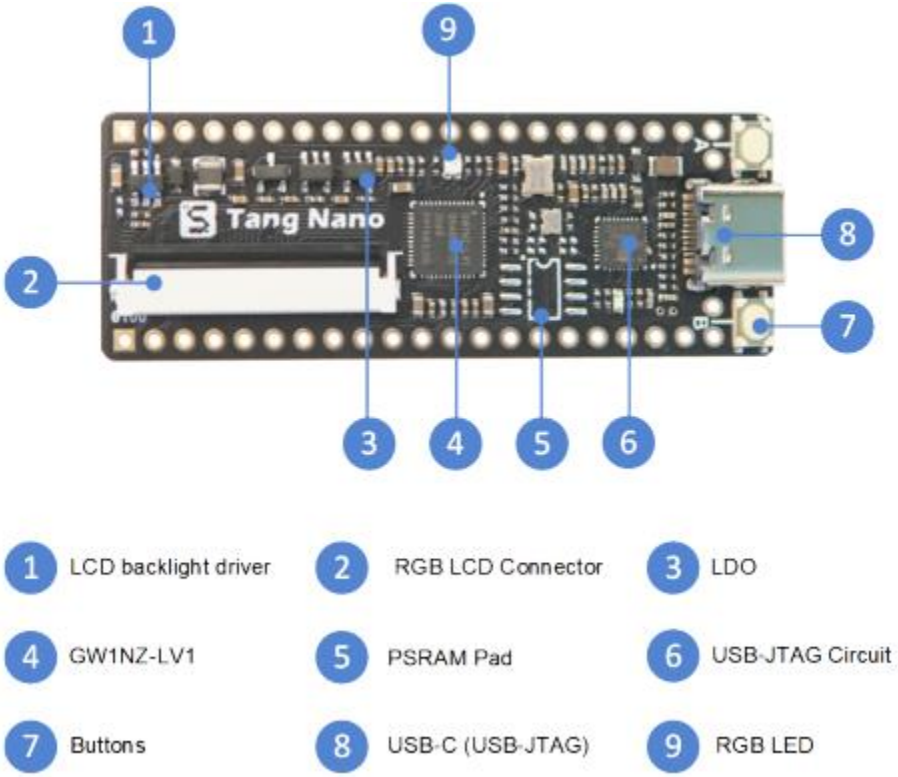


Figure 3.20 – Gowin Tang Nano 1K Diagram

5]2] FPGA Software:

By choosing the GOWIN FPGA, we will use the brand custom IDE software and it is the **GOWIN EDA (E**lectronic **D**esign **A**utomation tool**)** which is a free software with a size of 300 Mb available at their website using this link: https://www.gowinsemi.com/en/support/download_eda/

Figure 3.21 – Download Page from Gowin Website

The EDA supports both Windows x86/x64 and Linux OS, and it includes the program flasher and USB drivers for the computer and the FPGA itself.

License is free and renewable every 1 year, to Apply license you need to login to the Gowin website and click this link: https://www.gowinsemi.com/en/support/license/ after filling the forum with personal information (name, email, enterprise…etc.) and Computer MAC Address you will submit a request for a new license file that will be sent your email, with this file you can legally open and use the GOWIN software for completely for free.

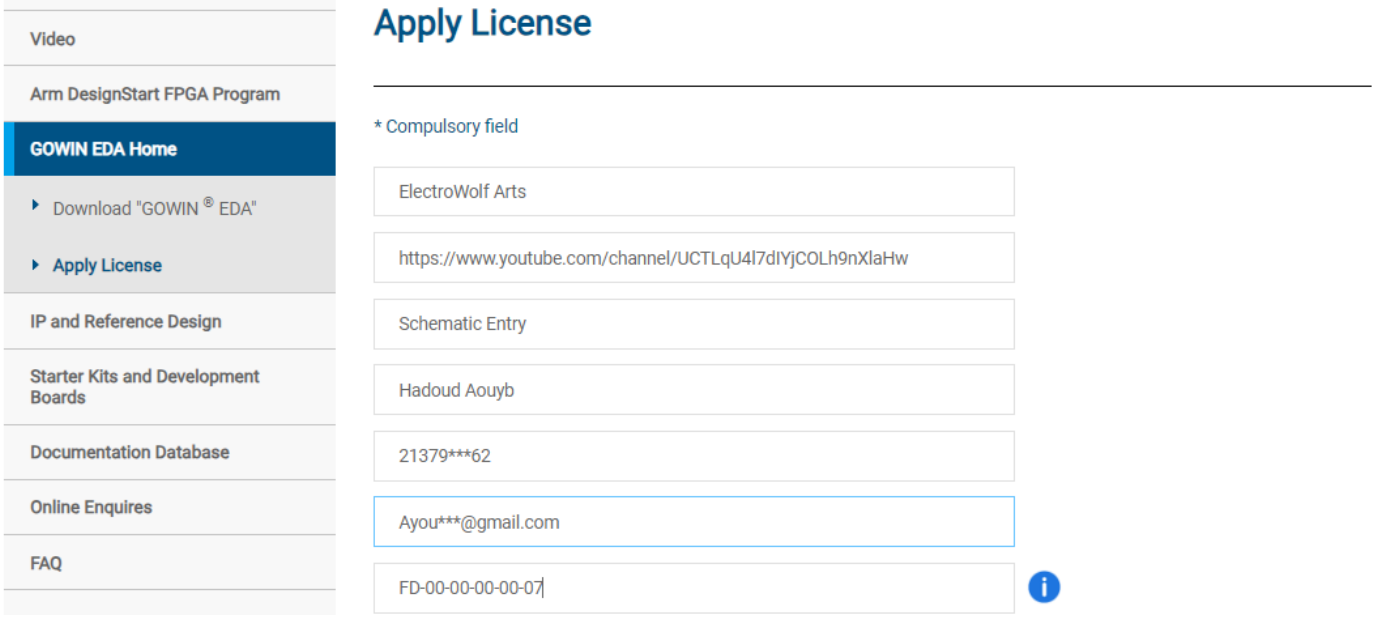


Figure 3.22 – Apply license Page from Gowin Website

5]3] Schematic Software:

If it happens for you to know how to program using VHDL or Verilog, you can proceed directly to create the project right away, but learning such languages is not that easy. In if fact, it can be more challenging compared to Python or C++ because they are hardware description languages primarily used for designing digital circuits, that's why there is Schematic Entry method which has very simple fundamentals.

Unfortunately GOWIN EDA doesn't built in schematic panel to draw our project circuit, so we are going to use third party software called Multism 14.1 from National Instruments, this software is a standard SPICE circuit design simulation software, just like ISIS Proteus but Multisim has a very unique feature which is **PLD design** (**P**rogrammable **L**ogic **D**evice) which is a part of the software that's allows to draw logic circuit and flash it directly to the FPGA.

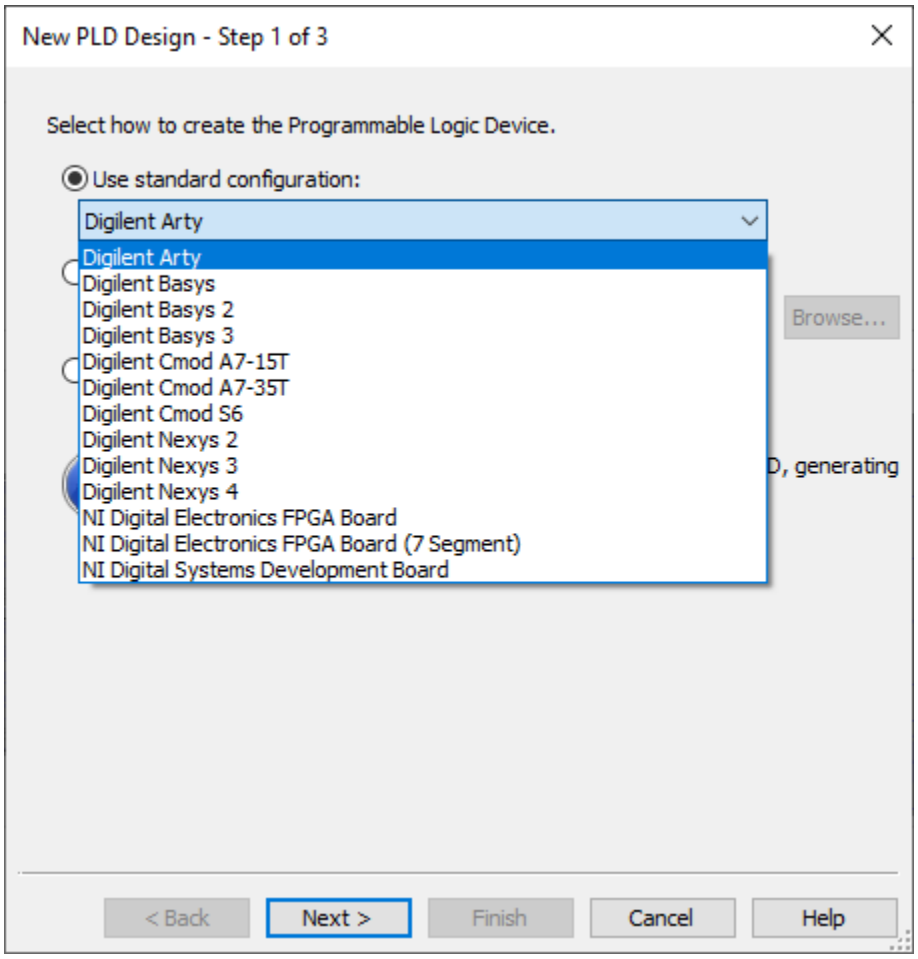


Figure 3.23 – Multisim New PLD Design window (New Project)

TangNano 1K FPGA is not included in the standard configuration because it's not commonly used as the **Nexys** or the **Basys**, but there is another option that solve the problem which is "Create Empty PLD", this option enables exporting the circuit that we draw into package files that will embedded in the GOWIN EDA later.

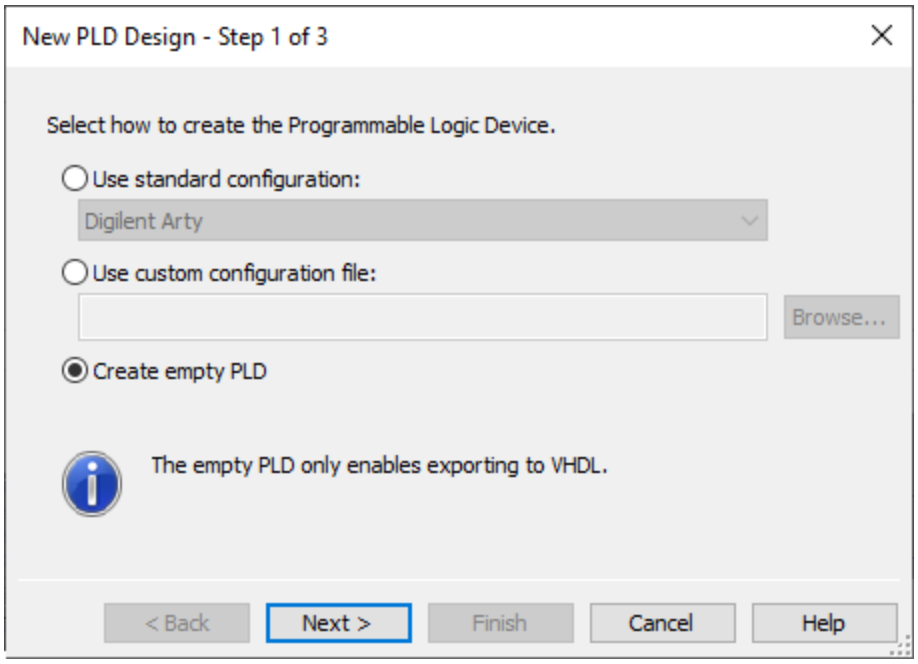


Figure 3.24 – Multisim New PLD Design window (New empty PLD)

Then we have to give a name to the project and it must match the name in the FPGA software:

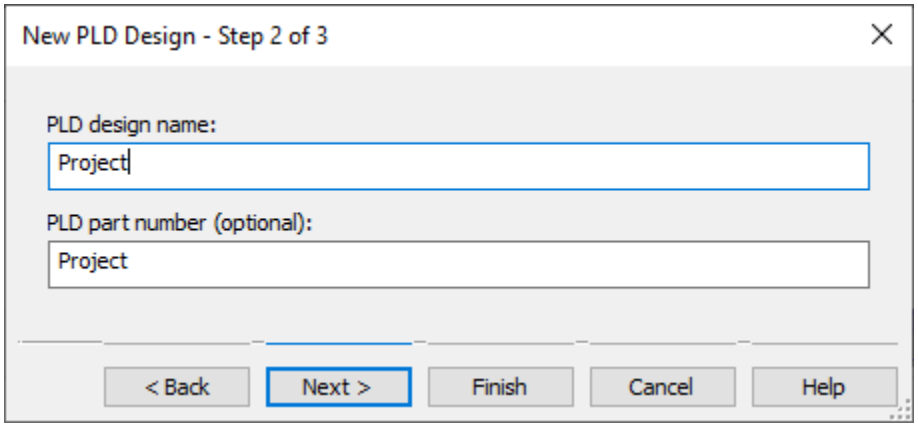


Figure 3.25 – Multisim New PLD Design window (Project Name)

After that we select the ports voltage, in our case the TangNano rates at 3.3 volt so we set that in this menu:

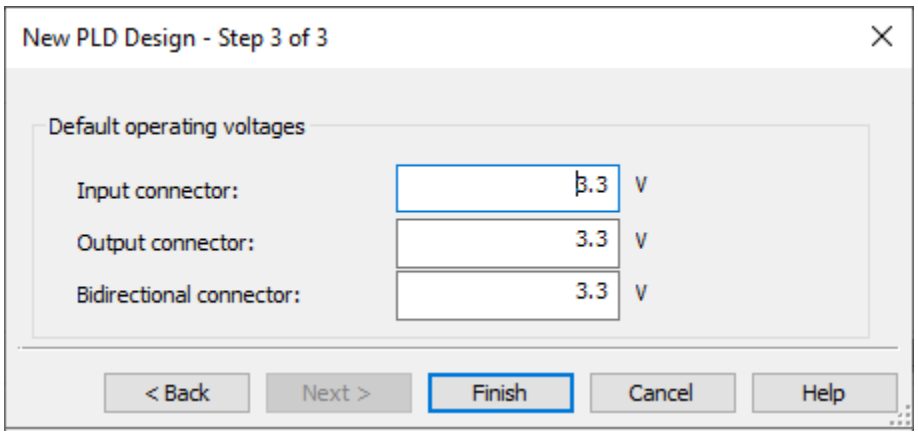


Figure 3.26 – Multisim New PLD Design window (Simulation Voltage)

Finally we press finish to open a blank sheet ready to draw our logic circuit.

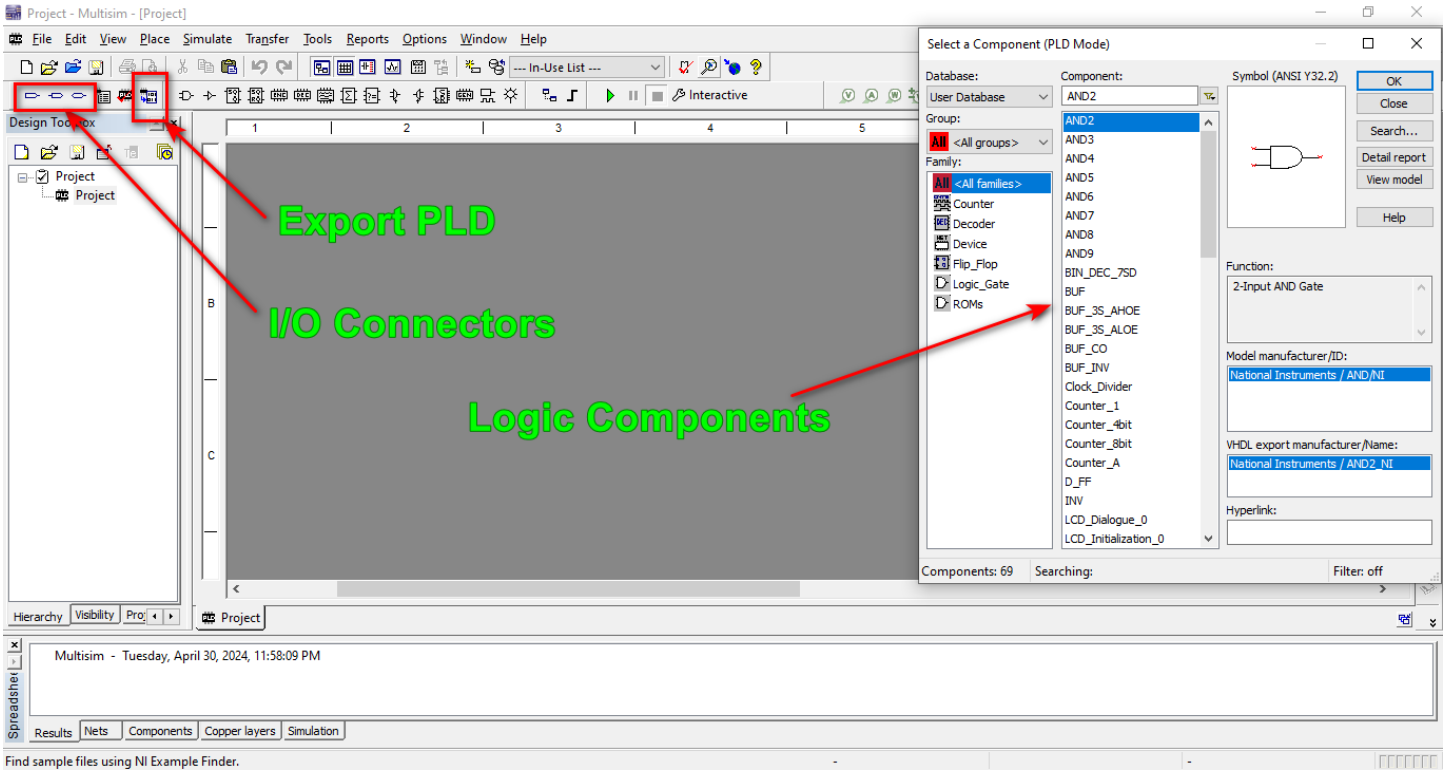


Figure 3.27 – Multisim window

By opening the components menu, we can select logic gates, Flip-Flops or other logic devices, we can also select Input/Output connectors and prebuilt circuits.

5]4] Creating the Project:

We will create something simple and useful, a 4bit digital counter which can be found anywhere (Clocks, Waiting Halls, Products Counter...etc.) to make this circuit we just have to use our knowledge in Logic binary. We know that a counter can be made out of D Flip-Flops so we drag it component 4 times since it's a 4bit counter. Then we connect the Output ~Q to the Input D for each one to get the switching effect of the JK Flip-Flop.

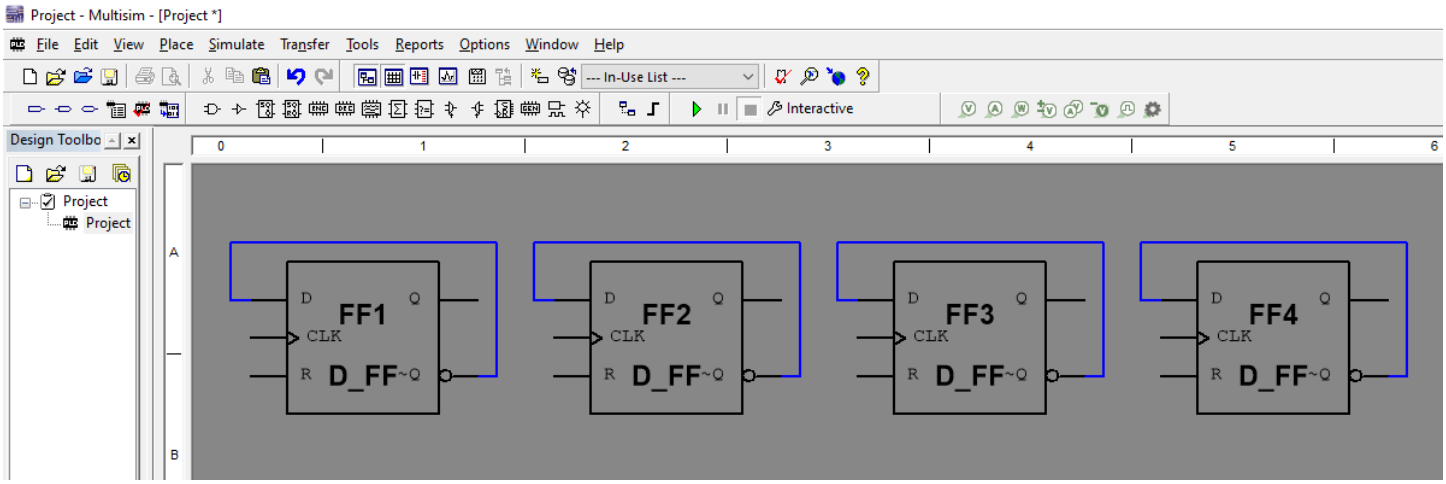


Figure 3.28 – Multisim window (Building a "Project" -D Flip Flops-)

Then we link all Reset pins together and connect them to Input Connector [R], to make the counter count upward we connect the CLK pin to the output ~Q of the previous Flip-Flop , and to make it count downward we connect the CLK pin to the output Q of the previous Flip-Flop.

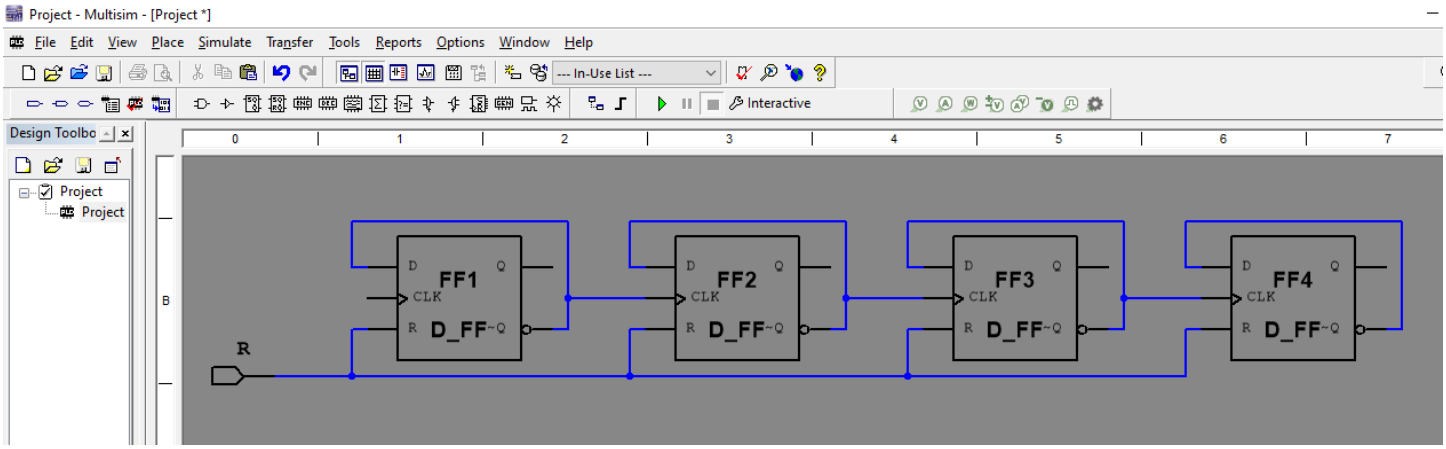


Figure 3.29 – Multisim window (Wiring the D Flip Flops)

After that we drag a BCD-to-7segmets Decoder from components list and connects its inputs (A0,A1,A2,A3) to the “Q” outputs of D flip flops respectively. And then connects the outputs of the decoder to output Connectors.

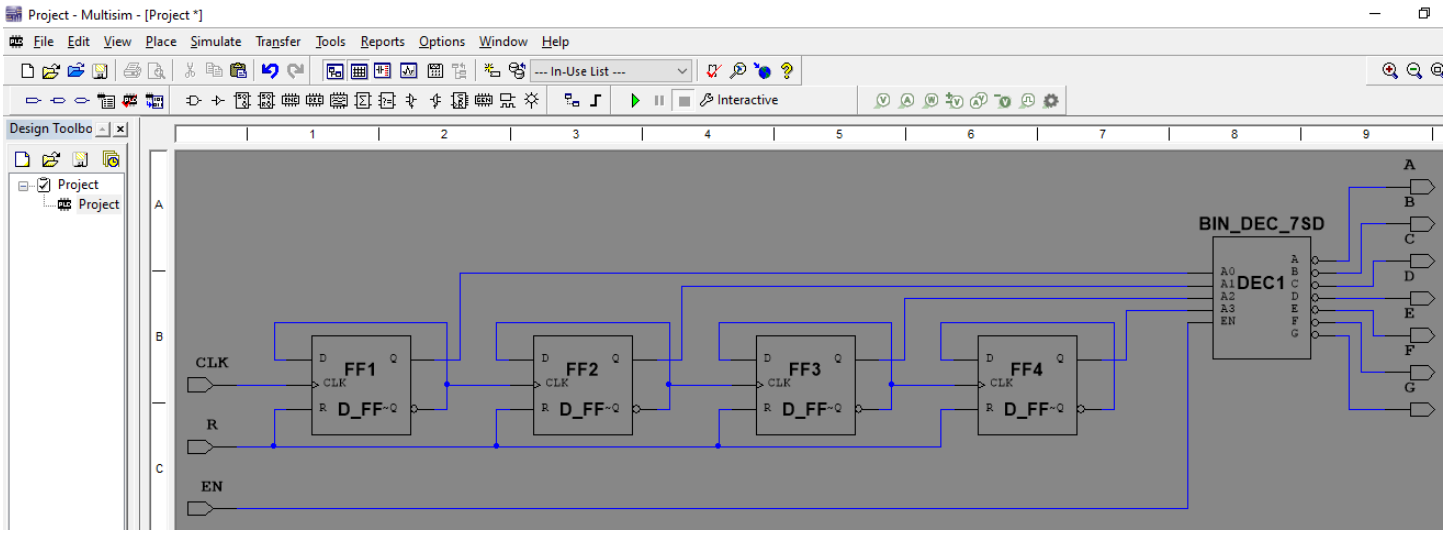


Figure 3.30 – Multisim window (Adding a 7segment Decoder)

Now to make the decoder works we just have to toggle it to “1” by the [EN] pin, and to make the counter runs we need to feed it a clock signal, we can do it automatically by connecting it to a clock generator or the onboard crystal oscillator, or we can do it manually by connecting a physical button to the **selected CLK pin** in the FPGA

Finally we verify opened (unconnected) pins, if everything is complete we press the “**Export PLD**” button on top-left and create the 2 package files that deliver wires routing, connections and logic components descriptive code.

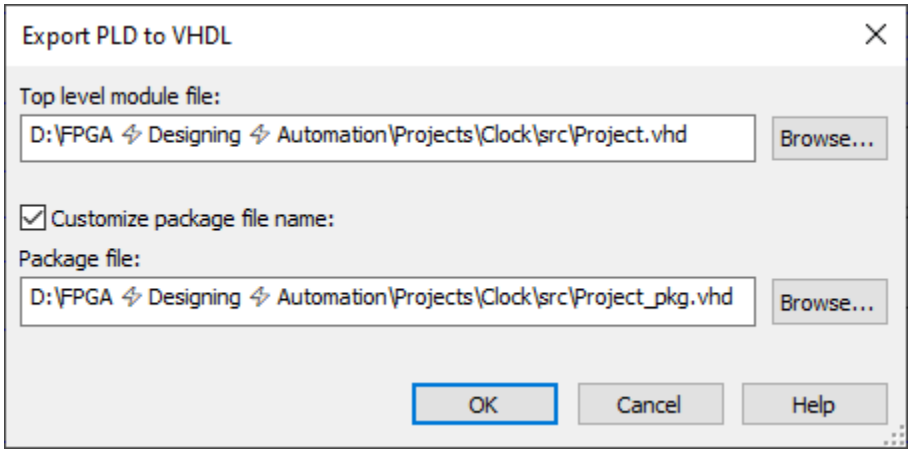


Figure 3.31 – Multisim window (Exporting PLD files)

5]5] Compiling and Configuration:

Next Step, we open GOWIN EDA software and create a new project with the same name that we typed in Multisim, and then select the FPGA we are working with (**GW1NZ-LV1QN48C6/I5**) and click “Next” then “Finish”:

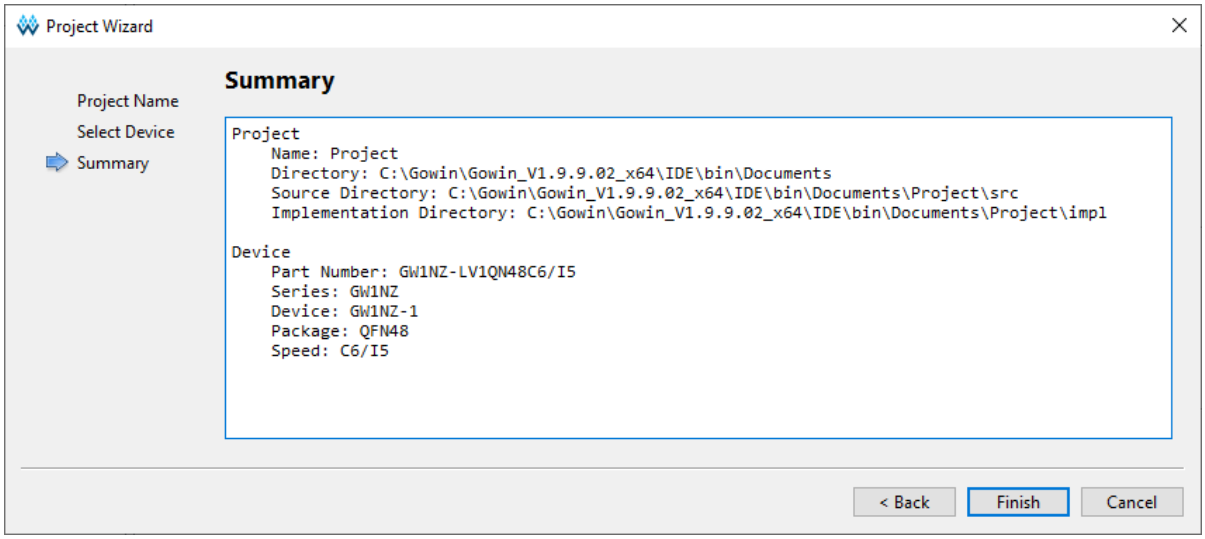


Figure 3.32 – Gowin IDE window (Project Wizard)

Then we select Project Design Panel and add new files which are the 2 files we generated in Multisim.

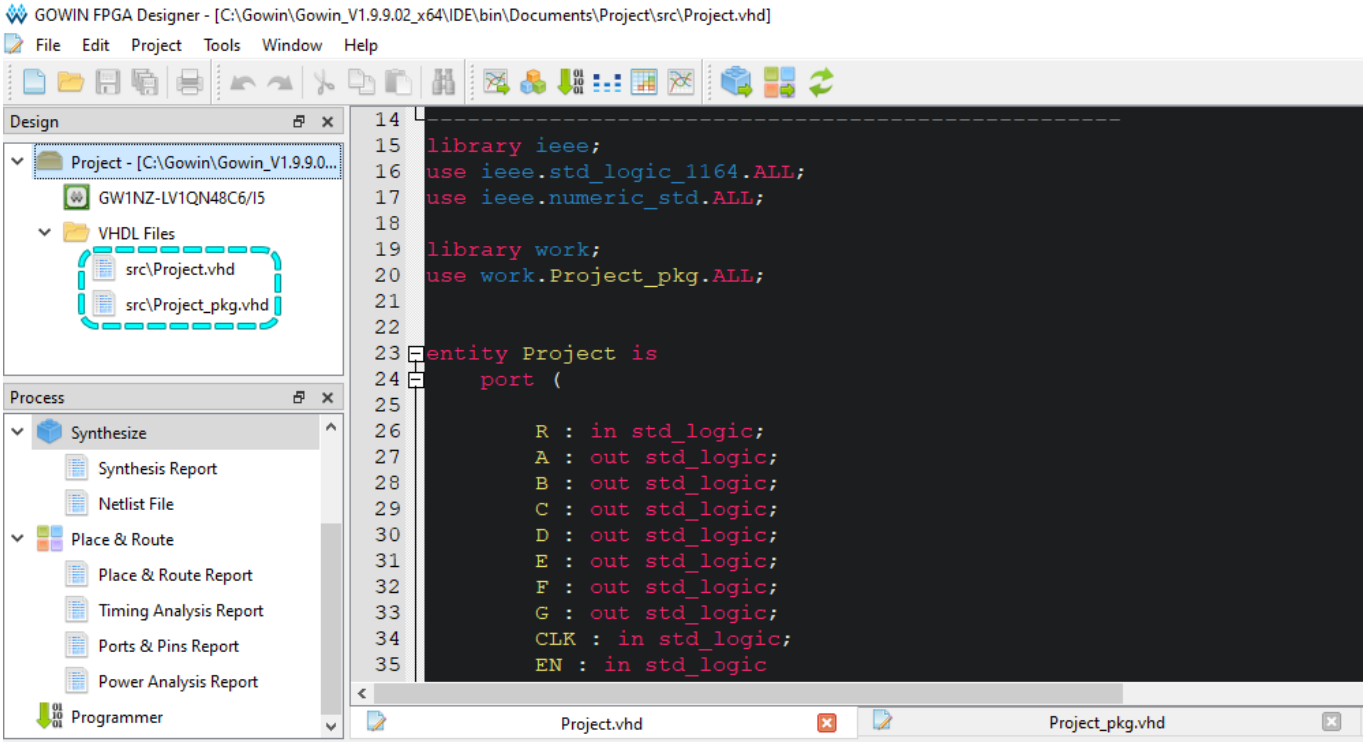


Figure 3.33 – Gowin IDE window (Uploading PLD Files

Now we create a “Physical Constraints File” which allows make the connections to the FPGA board physical numbered pins, we chose the right pins row for the decoder output for easier routing and the set the onboard buttons for reset and manual clock generating, then we assigned the [EN] to a random pin and connected it to ”1” (3.3V power pin).

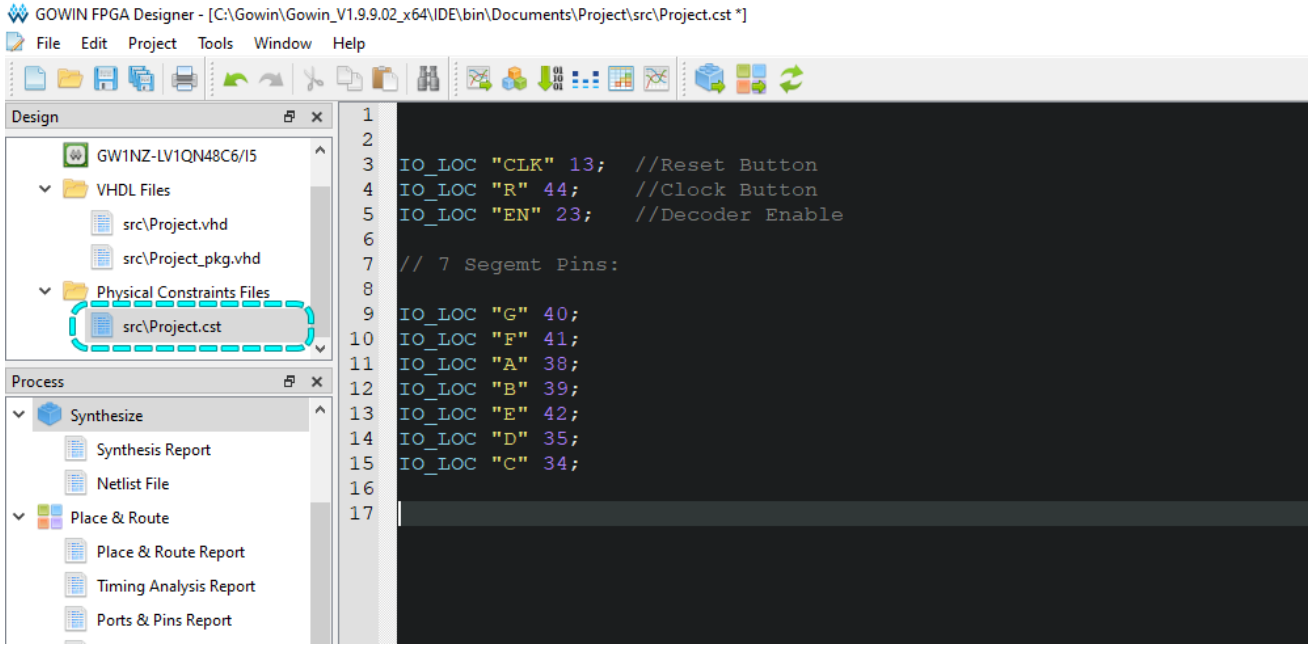


Figure 3.34 – Gowin IDE window (Creating Pins Connections)

After that, we move to the “Proccess Panel” and run the “**Synthesize**” report and “**Palce & Route**” report, and then we check the consol for any errors such as unconnected wires and connectors, typing mistakes or mismatched titles….etc. if there is no error , the **Bitstream** file will be generated.

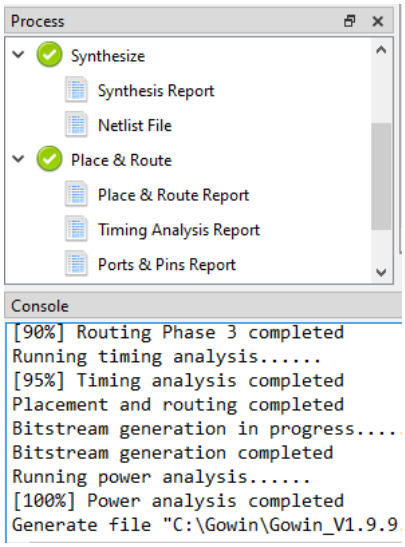


Figure 3.35 – Gowin IDE window (Synthesize report and Palce & Route Compiling)

Finally after creating the **Bitstream** which carries all the work we did from the beginning of this project, now its ready to be flashed to the FPGA. To do that we open the GOWIN Programmer, Select the “Operation” for storing the Bitstream file into a volatile memory or a external flash memory to embedded the file into the FPGA. And finish by clicking “Program/Configure” button:

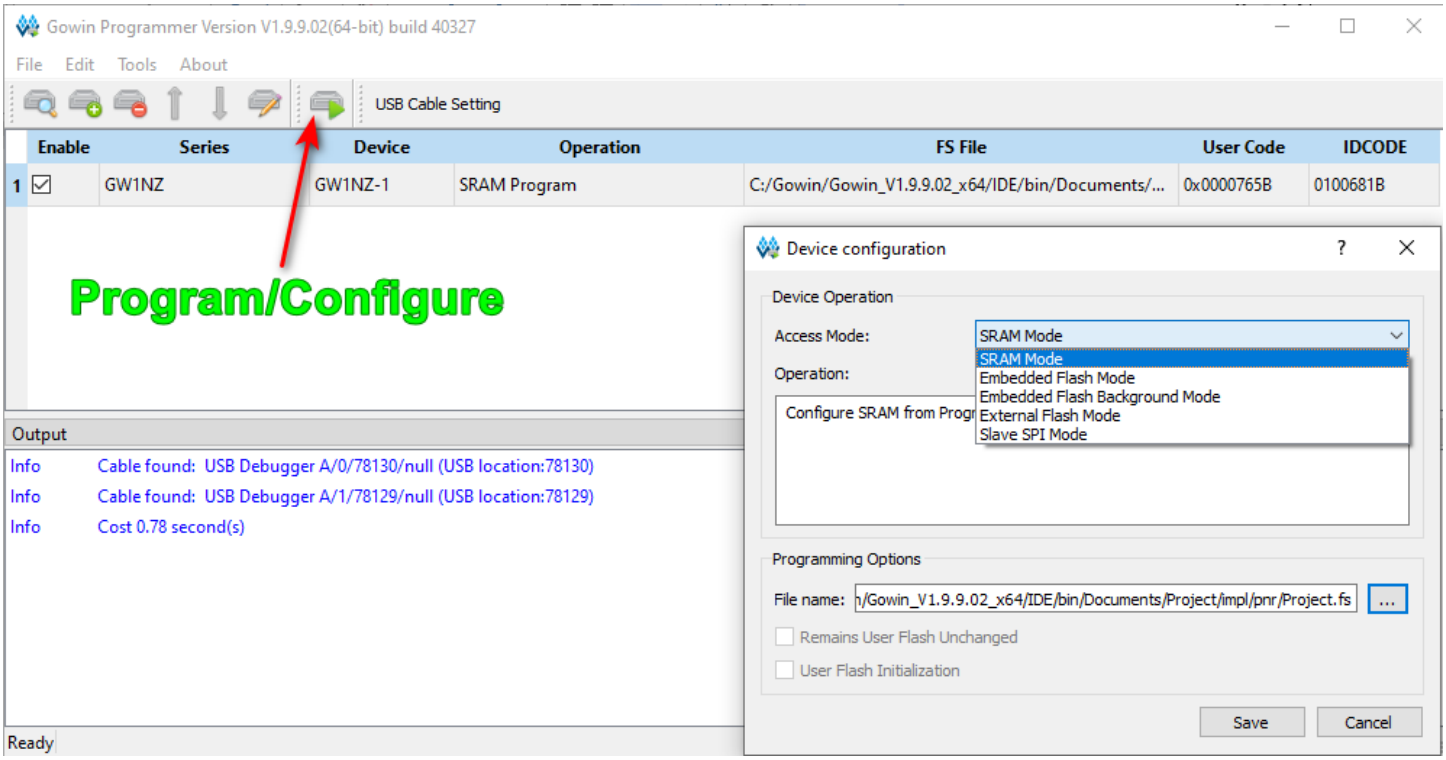


Figure 3.36 – Gowin Programmer window

The bitsteam file will be flashed to the selected memory and when it finishes, the FPGA will simulate our circuit insantly.

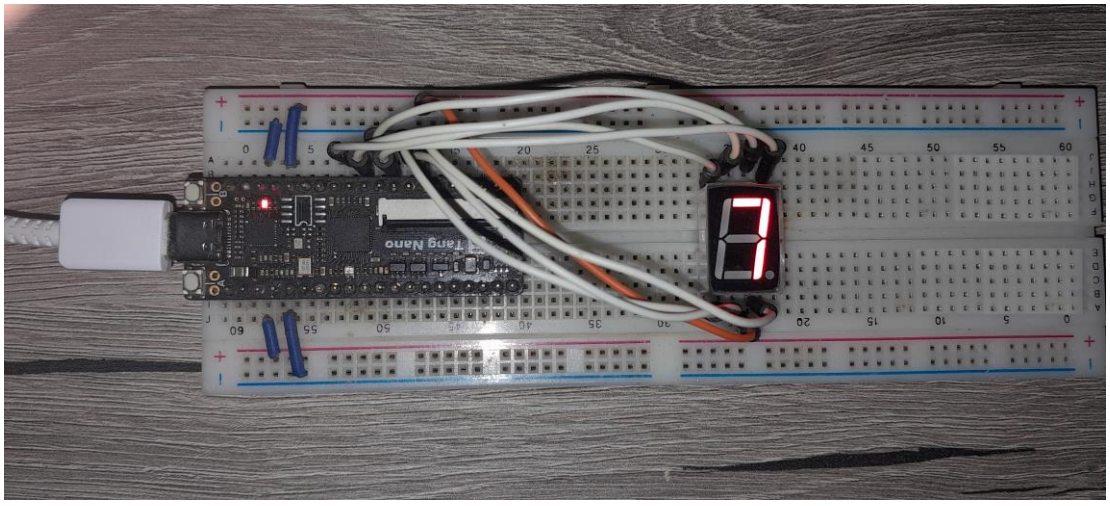


Figure 3.37 – Project Implementation Using the FPGA and 7segemnt Display on a Bread Board

Now we test our logic circuit and check for any mistakes, we can always correct the circuit using multisim and overwirte the files, or rebuild the whole thing.

This project is open for improvments, we could add a second 7SEG display driven by a squenced counter , or enhance the accuracy by switching to JK Flip-Flops.

The downside is that the TangNano 1K cant doesn’t have analog integrety to recieve and output variable signals due to lack of Analog to Digital Converters, so we are limited with digital project only.

-Realization-

“Auto Labeling System”

1] Project Preparation

After learning the basic of automation systems and how they operate in the industry field, understanding the role of microcontrollers, captures and actuators on these system and how they work together. Exploring the FPGA, so far the most uncommon controller known by developers and hobbyists and showing the main methods to configure it, such as VHDL/Verilog, C++, or Schematic Entry which is the goal of this whole work, the idea of creating an automated system without writing a code or even knowing how program is something unique itself, just by creating visual understandable logic circuits, it can't be easier than that.

Now we will test the FPGA in a real advanced task, given the specs of board we have, we can create a small simple automated system using digital interfaces since the GOWIN FPGA doesn't support analog. Even though, making a working automation replica of industry is not technically easy due to limited resources of materials and components, we don't forget that the FPGA itself was hard to get and there was few options to pick from.

So the plan is to gather as much e-waste components and parts from broken or abandoned devices to reduce the bill cost if some parts require buying.

The best example of automation is production line, it consists of many segments depends on the product complexity, quality and hardware. Of course there will be captures (sensors) and actuators and a controller to run and process the system.

We chose a simple segment of the production line which is product labeling, this part of the system can be positioned at the end of the line before packaging and it resides of **Conveyer Belt** and **Moving Printer**, as well as Motors, relays, LEDs and captures.

2] Sketching:

First, we draw a hand sketch to understand what we are working by visualizing the movement of the project and components positions. The sketching is tryout before prototyping and it may not be the final version.

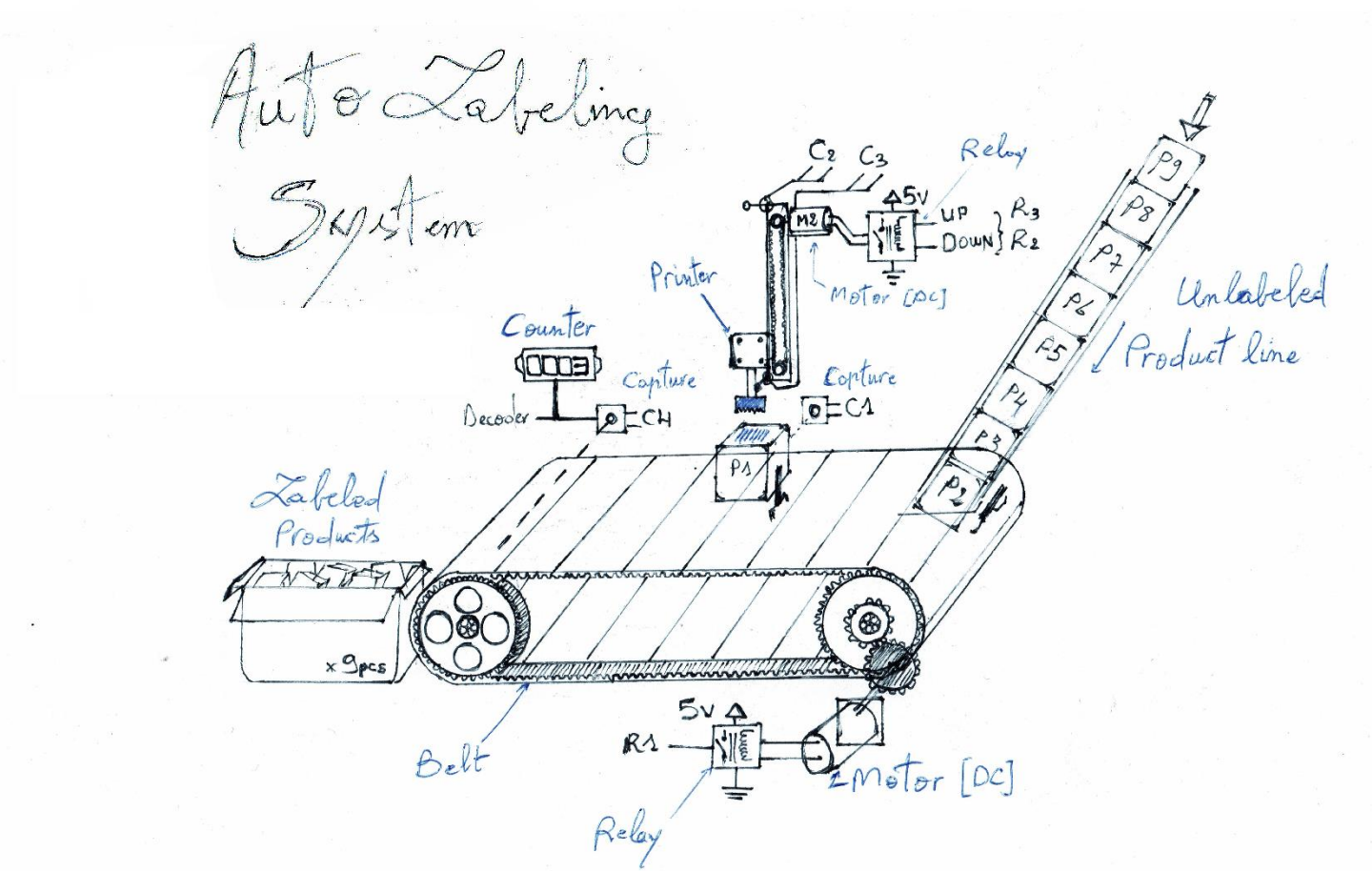
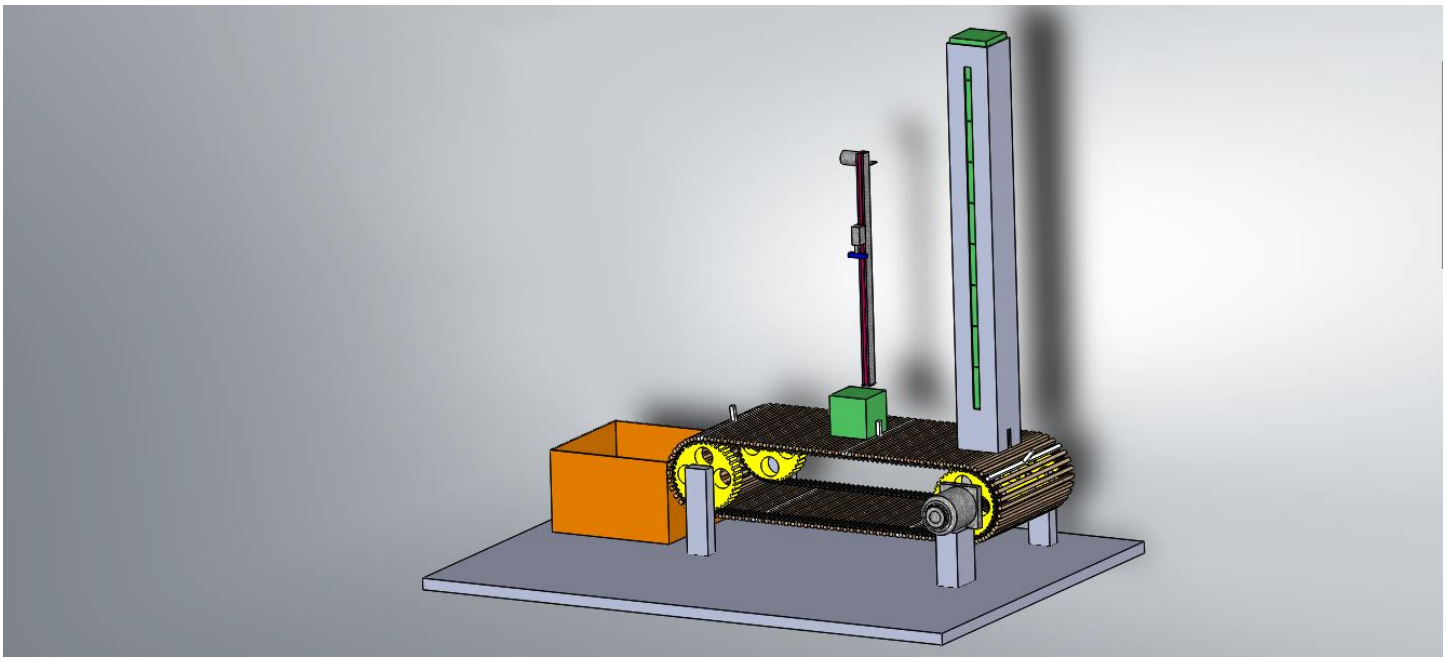


Figure 4.1 – Hand Sketch Represents the 1st Prototype of the Project

3] 3D Modeling:

In this step, we realize a 3D model of the project using a computer software like Dassult Systems **Solidworks**, the 360° view of the model makes the project visualization more understandable and clear. The 3D model will not be a 100% replica of the final project due to materials limitation.



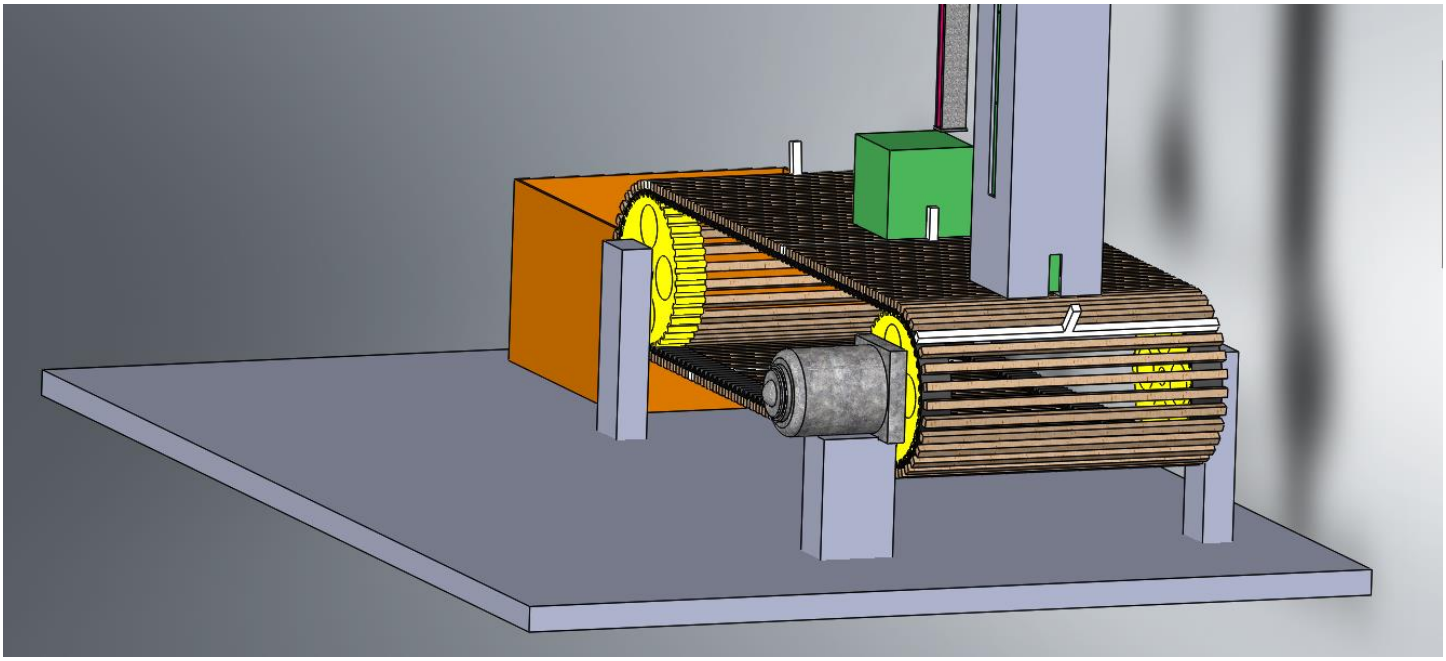


Figure 4.2 – 3D Assembly Represents the 1st Prototype of the Project

4] Materials list:

1. **Conveyer belt:**
   - 24V/2A Stepper motor
   - 2 x Pipes (117mm, ⌀ 20mm INT, ⌀ 25mm EXT)
   - 3 x OUTDO 608RS ball bearing (8mm x 22mm x 7mm).
   - 2 x wooden support stand.
   - Belt: Plastic sheet loop
   - 3 x Bolts (8mmx30mm) + 3 Nuts (8mm)

2. **Printer: The half of this part is pre-assembled and it consist of:**
   - 12V/2A DC motor
   - 2 x limit switches (using computer mouse switches).
   - 1 x IR sensor.
   - Timing belt.
   - 2 x gears.
   - Mounted ink labeling stamp.

3. **Product Dispenser** :
   This part is a square shaped pipe made of cheap material like wood or card board, and it have almost the size of the product and had a notch at the bottom so the product can be hooked out using the conveyer belt hook

4. **Control :**
   - GOWIN TangNano 1K FPGA board.
   - Stepper Motor Driver DRV8825
   - Quad Relay Module HW 316
   - Control buttons. (Start, Pause, Reset)
   - Power Supply

5. **The Frame**
   - The frame is like the chassis of the system consist of a Base to hold the all the parts and wood bars to hang the printer and the dispenser. As well as routing the wires for electric component

5] Electrical Circuit:

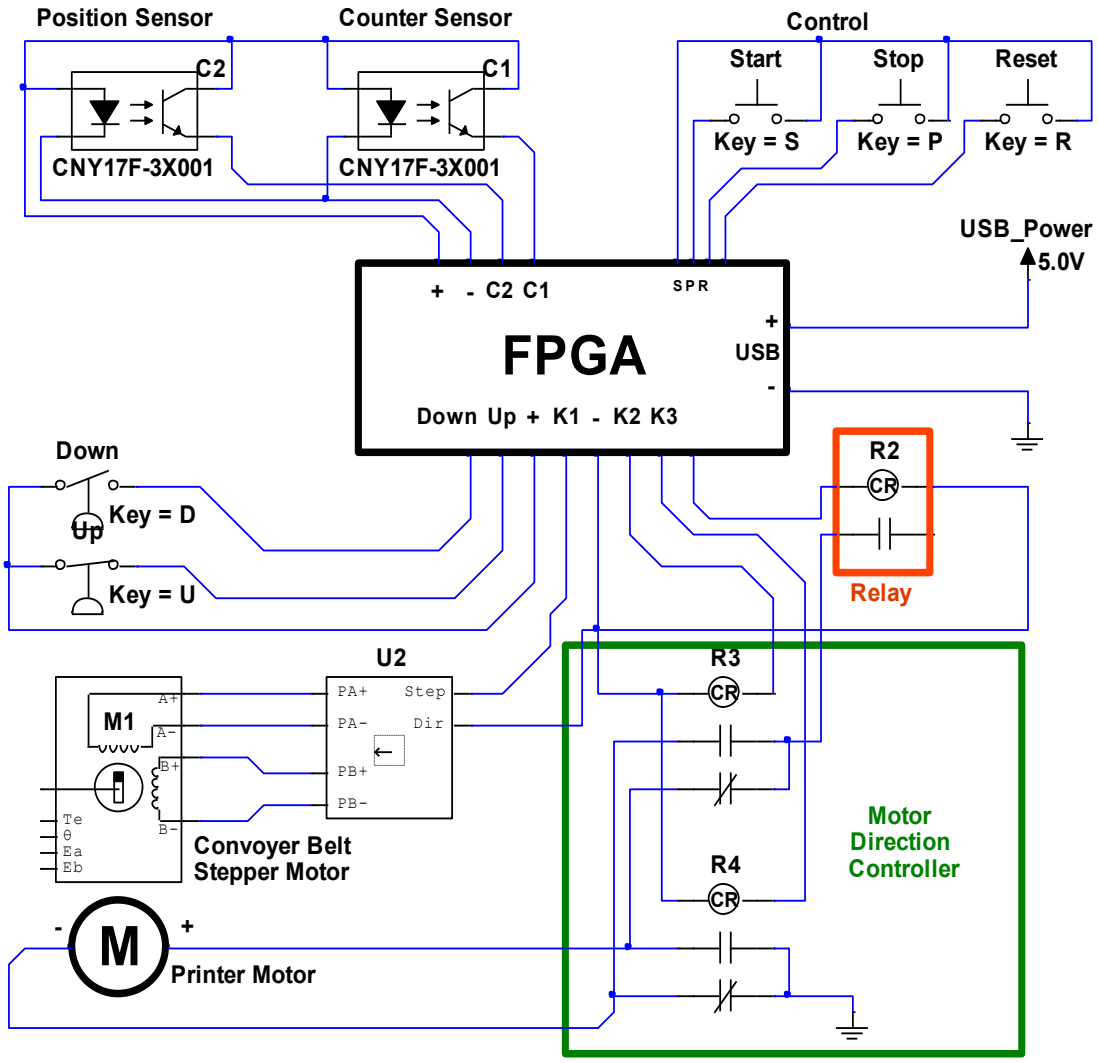


Figure 4.2 – The Main Electric Circuit for the project drawn by Multisim

Motor Direction Switch Circuit:

This circuit is made of 2 relays combination that control the current direction that flows to the printer DC motor to lift the mounted ink stamp up and down by providing a signal to the relay input.=

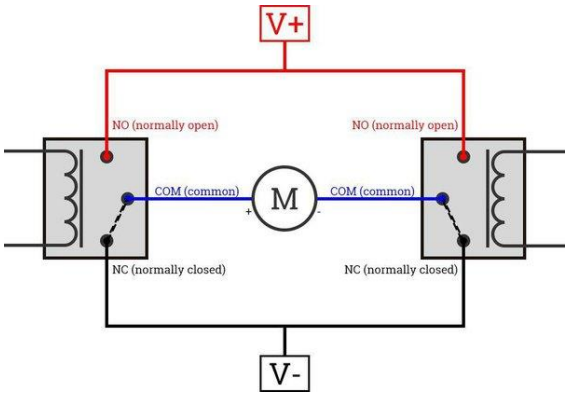


Figure 4.2 – H-Bridge Motor Driver

6] Schematic Entry Circuit:

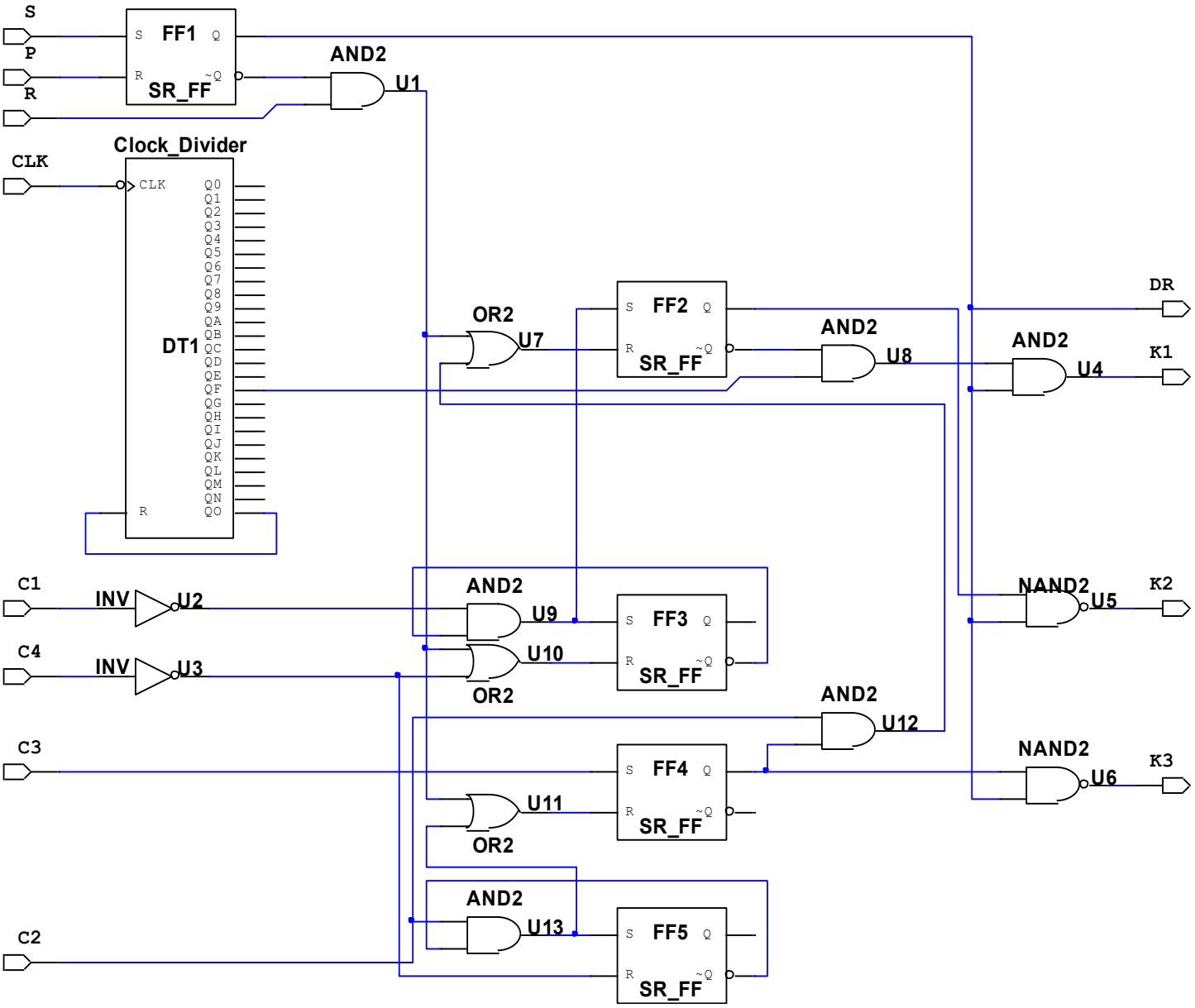


Figure 4.2 – Logic circuit of the Project designed and built in Multisim PLD Design

Conclusion

In conclusion, this project has demonstrated the feasibility and efficiency of a labeling system based on FPGA and schematic input, offering great flexibility and scalability. The superior performance and reduction in energy consumption make this technology attractive for high-demand applications. Additionally, the potential of this system paves the way for the creation of an innovative startup in the field of labeling systems. This project provides a solid foundation for future developments, integrating advanced sensors and communication modules.

References:

- Automation History (page-4): https://blog.sasken.com/industrial-automation-the-history-of-manufacturing-application-current-status-future-outlook
- Transistors History (page-25): https://www.pbs.org/transistor/album1/
- Verilog code for the AND Gate: https://circuitfever.com/logic-gates-verilog-code
- HLS code for AND Gate: https://www.geeksforgeeks.org/program-to-implement-logic-gates/
- Any other introductions, definitions, examples and schematic guides are created by the author of this graduate thesis
- Illustrated images and diagrams have been are created using **FastStone Capture** software and **Microsoft Paint** by modifying downloaded PNG pictures and sketch drawing.
- PNG images downloading using: https://www.pngwing.com/en/
- Circuit images are created by and captured from **Multisim** software.
- Gowin Semiconductor website: https://www.gowinsemi.com/en/
- TangNano 1K FPGA Datasheet and User guide: https://wiki.sipeed.com/hardware/en/tang/Tang-Nano-1K/Nano-1k.html
- National Instruments Multisim Software download: https://www.ni.com/en/support/downloads/software-products/download.multisim.html#452133