Appunti DESD

Riassunti delle lezioni

Luca Daidone

19 giugno 2023

POLITECNICO
MILANO 1863

Indice

Elenco delle figure

1 FPGA - Structure and Logic

FPGA are digital devices that have 3 types of resources:

- **Logic Blocks**
- **I/O Blocks**
- **Programmable interconnects**

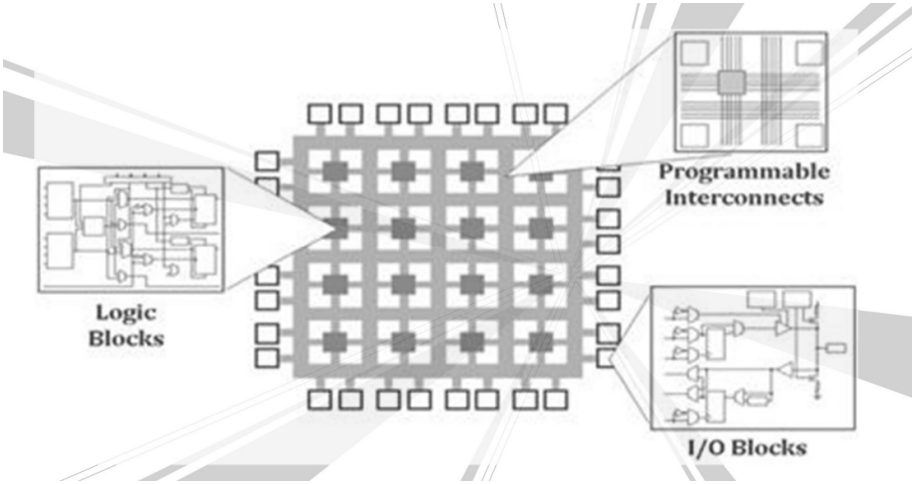


Figura 1: FPGA structure

The concept of total reconfigurability is very important in FPGAs. It can be performed even during run-time.

1.1 LUT

One of the most important blocks in an FPGA is the **Look Up Table**, LUT, which is basically a memory. LUTs are used to synthetize functions or as a memory blocks. The address bus of our LUT represent the input of the function and in the cells addressed by the inputs we store the result of our function.

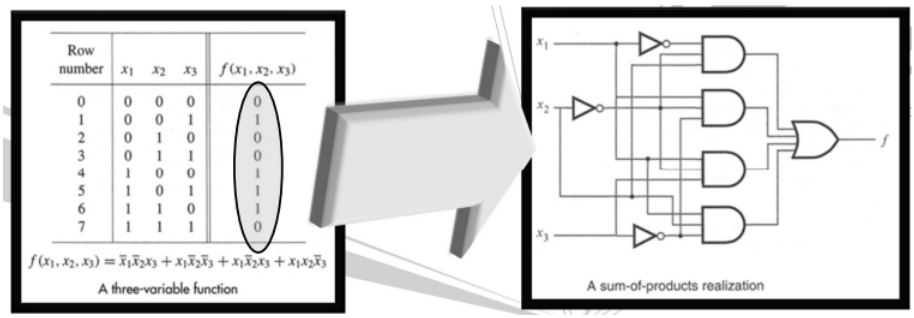


Figura 2: LUT

1.2 I/O Resources

These are very complex resources. There are several features that can be changed on the I/O blocks.

- Tri-state control
- Flip-flop/latch for timing optimization

- Pull up/down resistors
- Controllable voltage levels

The configuration bits set all the feature we want on the I/O ports.

1.3 Interconnection resources

These are a see of islands connected by an array of connections. A single node of this array is made by several transistors, such as the connection can be customized in many different ways. There are millions of nodes. The lines are real conductors hence they carry parasitism, therefore as the path becomes longer the RC effect increases.

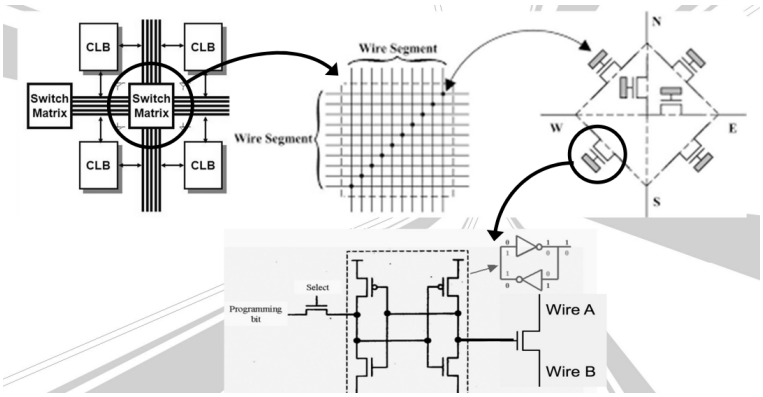


Figura 3: Nodes

To increase performance of a system, we don't work on speed but on architecture.
There are **local** and **global** interconnect lines, which are different.

1.4 Power

There are many voltage rails in a FPGA: 1.2V, 1.5V, 1.8V , 2.5V and 3.3V. Didn't say much more :(.

1.5 Configurable Logic Blocks - CLB

These are the block that are connected by the interconnection matrix. Each one of these has the following structure:

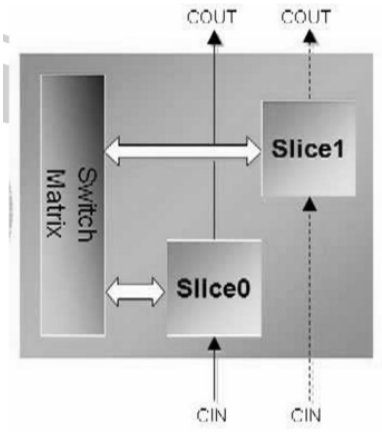


Figura 4: CLB structure

Since the base operation is the sum, we have **carry** in and out lines. The carry must be propagated in a very fast way.
The two slices are **SLICEL** logic slices and **SLICEM**. 60% of slices are SLICEL and 40% are SLICEM. In slicel the LUTs are used as logic elements while for slicem the LUT are used as memories or shift register mainly, even tho they can be used as logic elements as well.
General architecture of a slice is:

- 4x 6 inputs LUTs
- 8x storage elements
- Wide functions multiplexers
- Carry logic

Some slices support also data storage through distributed ram and 32-bit shift registers: such slices are the SLICEM, all the others are SLICEL. Each LUT in a SLICEM can be configured as logic, distributed ram or as a shift register.
Each CLB can contain either 2 SLICEL or a SLICEL and a SLICEM. Every CLB has 2 SLICEs.

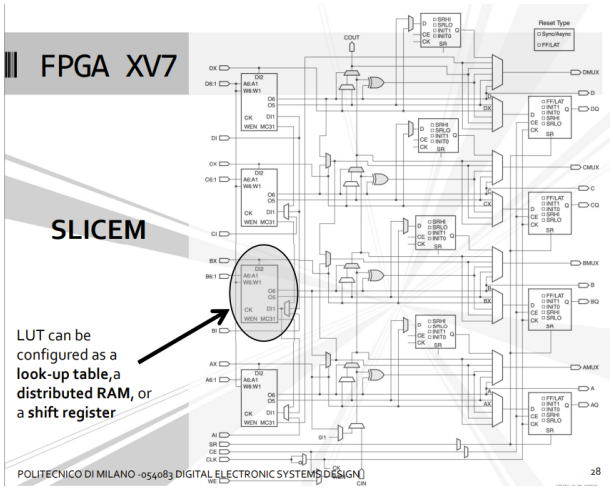


Figura 5: SLICEM structure

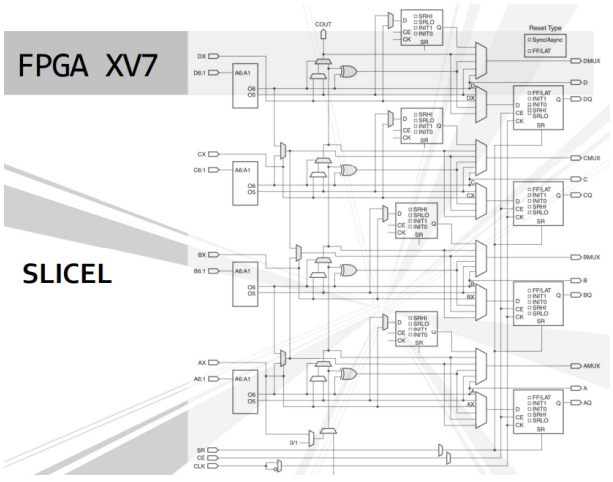


Figura 6: SLICEL structure

Every LUT can be used as a single 6 inputs two outputs function generator or as 2 function generators with 5 (shared) inputs and one output, or every combination with fewer inputs.

The 3 multiplexers (2x 7 inputs, 1x 8 inputs) can combine these function generators and create functions with 7 or 8 inputs in a single slice.
For functions with more inputs, more than one slice is needed, but this must be done between CLBs since there's no direct connections between slices within a single CLB.

1.6 Storage

There are 8 storage elements in a single slice. 4 can be configured only as edge triggered D FF the other 4 as either as edge triggered D FF or as level sensitive latches.

- The LUTs in a SLICEM can be configured as **ditributed RAM**, several LUTs can be organized to create larger RAM. It can be **single port** where write and read share the same address port, **dual port**, one port for synch and asynch write and one for asynch read, **quad port**, one port for synch and asynch write, three ports for asynch read.
- The other type of memory is block ram **BRAM**, dedicated RAM blocks of normally 36kb, that can be used as a true 2 ports RAM.

LUTs in a SLICEM can also be used as shift registers, they can be combined to create a max 128 clock cycles delay in a single slice. Of course different slices can be combined to create longer delays.

1.7 Carry logic

These are dedicated lines for carry, in order to perform fast arithmetic. We don't need to know this mechanism. MAYBE I SHOULD STUDY IT ANYWAY AAAAAAAAAAAA

1.8 Interconnections Resources

CLBs are placed in an array in the chip. Every CLB has an internal connection matrix which is connected to the **interconnection matrix** that runs along rows and column of the CLBs matrix. Other blocks can be connected using this interconnections lines, like BRAM and I/O block. Connections can be long and carry high capacitive parasitism.
A technology used is the **SSI** Stacked Silicon Interconnect. This technology is used to create **SSL** Super Long Lines, that connects **SLRs** Super Logic Regions. It is basically a technique to efficiently connect distant areas.

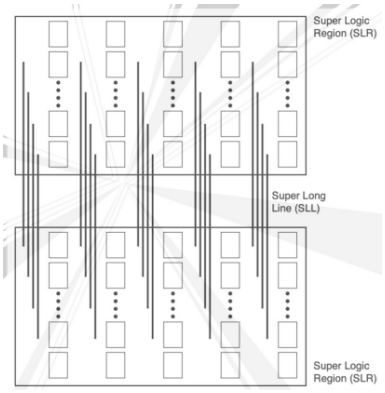


Figura 7: SLL

This methods allows us to use the "divide et impera" philosophy, where different modules that solve smaller problems are interconnected.

Only few interconnection resources are available to the user, like **clocking routing resources**, through clock buffers, **global set/reset** for the registers and flip flop, **global tri-state** to force the I/Os to high impedance. For general interconnection, the user can only use **floorplanning**: giving routing constraints to the synthetizer.
Cascading, like for carry logic or RAM block, is constrained within SLRs and can't be performed between them.

1.9 I/O Resources

To transfer data inside and outside the chip, buffers are used:

- BUFIO
- IDELAY/ODELAY to add delay to an I/O signal
- IBUFG clock capable input buffer
- IBUF/OBUF input output buffers

1.10 Packaging and Speed-grade

The package can limit the performance, it's extremely important. Geraci didn't say anything more LOL.

1.11 DSP Resources

DSPs satisfy the need of heavy mathematical processing. There aren't many, 300/400 per chip generally.

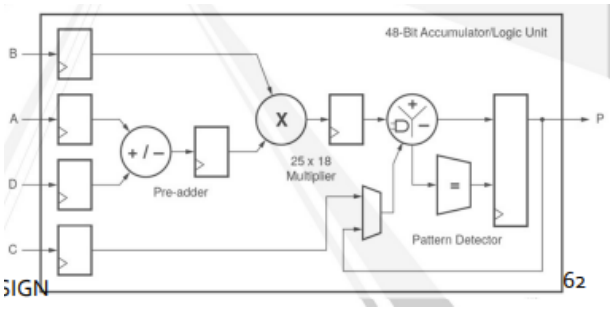


Figura 8: DSP Structure

Newer chips have also micro-controllers or microprocessors inside, since different applications require different solutions. These are called **SoC System on Chip**.

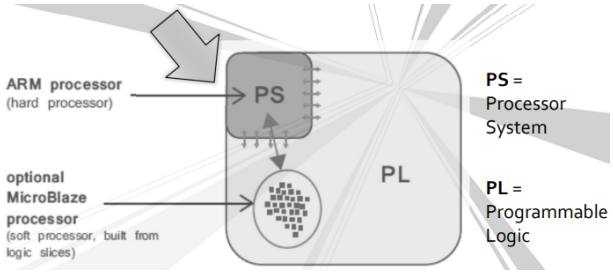
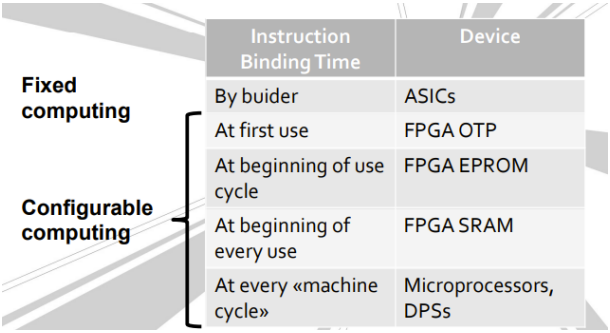


Figura 9: SoC structure

2 A New Paradigm: Fixed vs Configurable Computing

FPGAs, uC, uP, DSPs etc use the concept of re-configurability as opposed to ASICs, where functions are decided by manufacturer and can't be changed.
The **instruction binding time**, meaning the moment in which a function is selected and executed, can be changed by users.



Instruction Binding Time	Device
By buider	ASICs
At first use	FPGA OTP
At beginning of use cycle	FPGA EPROM
At beginning of every use	FPGA SRAM
At every «machine cycle»	Microprocessors, DPSs

Figura 10: Instruction Binding Time

2.1 Cost

For a mass scale production, FPGAs cost more per piece compared to ASICs. For this reason FPGAs are used in dedicated application, where few pieces are needed and re-configurability is crucial. SRAMS FPGAs have been used for example for the rovers on Mars, since they allow re-configuarbility of the chip on the field, meaning redundancy. In FPGA redundancy is attained by repeating the configuration of the FPGA every few hundreds of ms.
OTP FPGAs, where fuse are blown and re-configurability is not possible, are still used for high radiation environments.
ASICs have high **NRC non recurrent costs**, while low cost per die. You must pay for the design, EDA tools, DFT (design for test), foundry, yield & manufacturing loss etc.

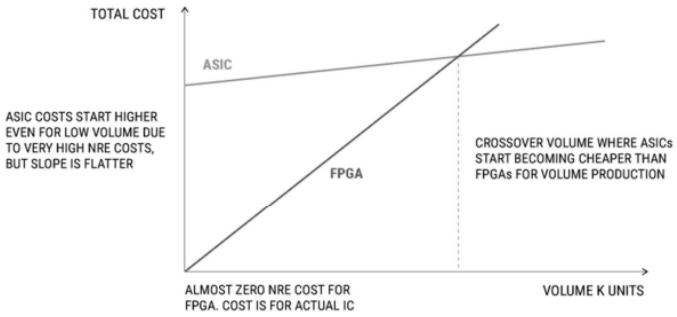


Figura 11: Cost

FPGAs start to reach the cost of ASICs for high volume production.

«ASIC suited for very high volume mass production, FPGA not»

ASICs are used when very high performance are required, that must be tailored on silicon, such as RF or very low power applications while FPGA are very handy for prototipation, also of digital ASICs.

2.2 Time to market

Another dramatic advantage is the time it takes to implement an architecture, which is days as opposed to ASICs where it's months.

2.3 Speed

FPGA are intrinsically slow since configuration is made through switches that carry parasitics effects. Therefore:

«Speed through architecture and not through clock»

Bottom line is that ASICs are faster anyway, but we try to patch this problem by developing smart architectures in FPGAs.

2.4 Power

We want to design for low power not only for power saving, but first of all because:

«Low power means high reliability»

since reliability is directly proportional to temperature and lower power means lower temperatures. Power consumption depends on device architecture.

2.5 Temporal vs Spatial

Temporal is when instructions are executed one after the other, meaning serial computing. In FPGAs the design is mapped onto the silicon therefore functions are "executed" in a parallel way.

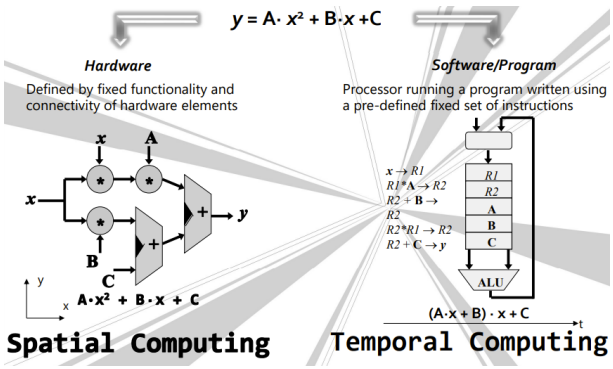


Figura 12: Serial vs Parallel computing

FPGAs are intrinsically slower but thanks to parallelization functions are performed faster.

2.6 *"Think parallel"*

The concept of **temporal parallelism** used in FPGAs produces **pipelined structures**. These are at the base of our designs.

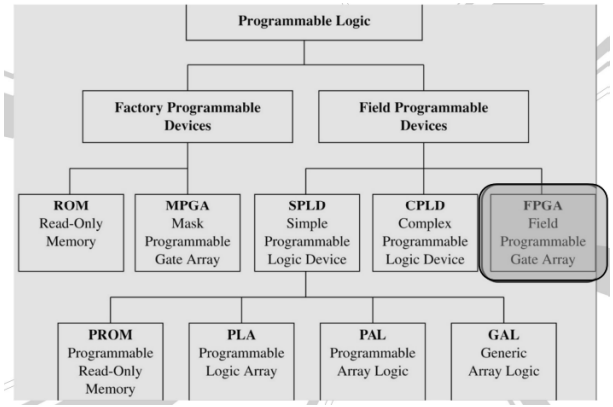


Figura 13: Programmable logic family tree

3 Clock and Timing

Every part of our systems obeys to the **clock**. However there are two types of systems: synchronous and asynchronous. The best performance are attained by asynchronous systems since there's no need to wait for the clock but we deal with synch one.
We work on **clock edges** which are affected by physics phenomena that modify them:

- **Jitter**: variation of the distance between edges (high f)
- **Wander**: drift in time of the distance between edges (low f)
- **Skew**: due to finite propagation time, the clock might take different times to reach two targets. This difference in time is the skew.

The clock is very critical therefore the distribution network is crucial:

«We will work with GHz» .

3.1 Clocking overview

There are global and local clocking resources. The devices are divided into **clock region**. Each clock region consist of:

- **Clock management tile CMT**: contains frequency synthetizers, deskew and jitter filtering blocks
- **All the synchronous blocks**: CLBs, BRAM, DSP, I/O banks

A region controls one full I/O bank and has an horizontal clock line **HROW** to which 50 to 100 CLBs are connected (in 7 series Xilinx devices). The global connection matrix to which CLBs are connected is not suitable for the clk signals, a dedicated line is needed. The philosophy is to have a low frequency reference and create only locally the high frequency clock.

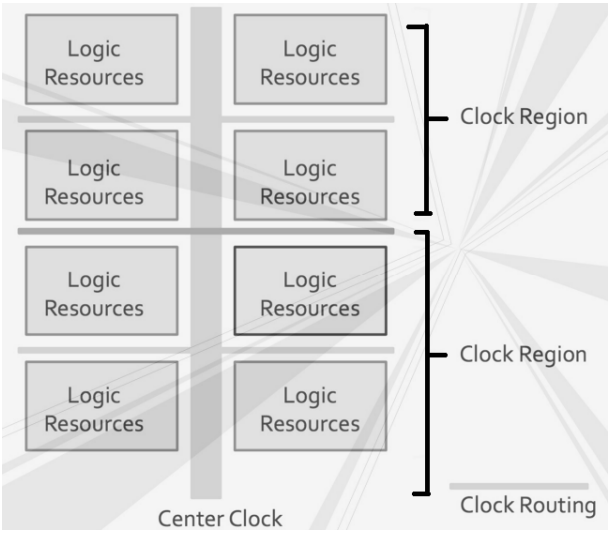


Figura 14: 7 series clocking overview

Below 1GHz the problems of skew and jitter are not important. Because:

«We will work with GHz» .

3.2 Clock Routing Resources

The I/O banks have a clock capable input, which brings an user-provided clock internally into the chip. Such clock is fed to a **global clock buffer BUFG** and to all the blocks of the clock region to which we have fed our clock, like CMTs and regional clock lines.
The **BUFG** is an evoluted buffer and can be used to:

- enable/disable a clock in the whole chip
- implement **glitch free multiplexers** since the BUFG can switch between clock sources
- implement compensation for clock delay when driven by a CMT

32 global clock lines are present inside the chip and the BUFGs are used to drive these lines.

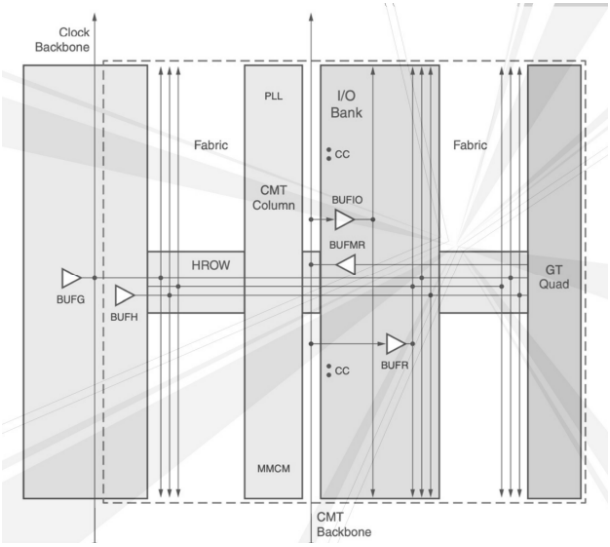


Figura 15: Buffers architecture

The other type of buffers are:

- **BUFMR multi-clock region buffer** is used to transmit a clock source among up to 3 adjacent clock regions.
- **BUFR** controls a single clock region.
- **BUFH horizontal clock buffer** are local for a single clock region and allow to interconnect the global clock lines to the local *horizontal clock lines* only. Each region has 12 horizontal lines so it can supports up to 12 different clocks.

- **BUFIO** clocks all the resources in a single I/O bank.

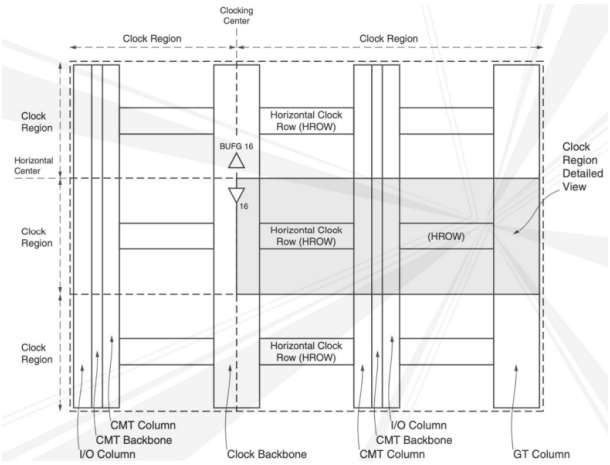


Figura 16: Clocking architecture

3.3 CMT Overview

7 series FPGAs have around 24 Clock Management Tiles, each of one consists of:

- **MMCM Mixed-Mode Clock Manager**
- **PLL Phase Locked Loop**

which are both used as frequency synthetizers, jitter filters and to deskew clocks as seen before. The PLL is a simpler version of the **MMCM**, in fact the latter:

- has infinite fine phase-shift capability in both directions,
- dynamic phase shifting during operation
- fractional division and multiplication for finer granularity of out frequency

The PLL are needed anyway to keep price lower, since having only MMCMs would increase the price too much.

«Make things simple, not simple things» .

3.4 System Timing

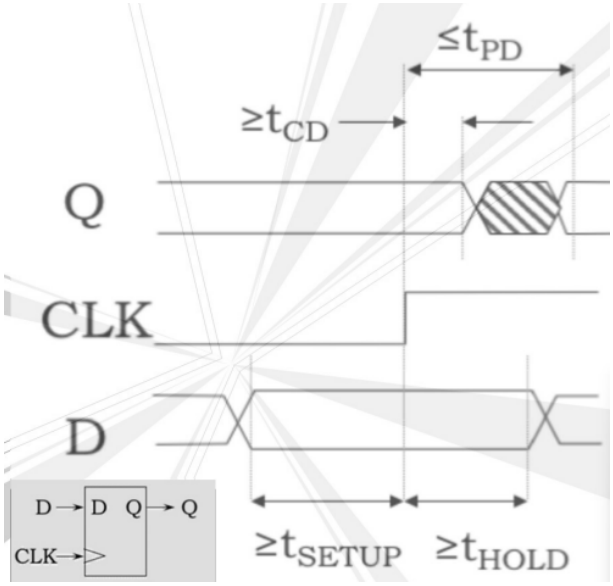


Figura 17: Timing requirements

Flip flop is the prototypical synchronous component. There are:

- Propagation delay: time between clock edge and valid output
- Contamination delay: time in which previous value stays constant after clock edge
- Hold time: time in which input must be stable after clock edge
- Setup time: time in which input must be stable before clock edge

The constraints we must respect:

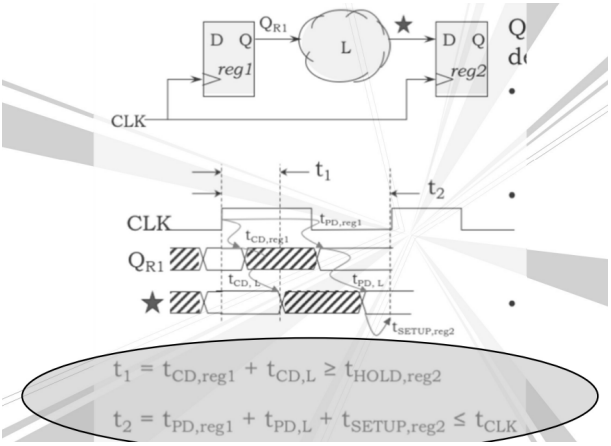


Figura 18: Timing limits

Clock period is limited by propagation delay of the FF and of the logic and by the setup time of the destination FF, or by the sum of the contaminations delay along the path, which must be greater than the hold time of the destination FF. Sometimes it's necessary to introduce gates to increase the contamination delay, that's also a reason why we say these systems are not optimized.

4 I/O Resources

«I/O are the harbor of the FPGA»

Output connections are done through I/O banks, that can be of two types:

- **HP High Performance** I/O banks. They maximize speed of communication, therefore use low voltages such as capacitive effects are reduced. Drawback is that they support only low current driving
- **HR High Range** I/O banks. Higher voltage, 3.3V, significant current driving capabilities hence high capacitive load driving.

4.1 Electrical Features

The 7 series FPGAs have **SelectIO** drivers and receivers. These are blocks that allow to set many different standard interfaces (LVCMOS, LVTTL etc.) on the I/O pin, to control the output strength (in mA), termination impedance, pull up/down, to introduce delay, to control slew rate and to set internally generated reference voltages. A bank is made of 50 pins and apart from the first and last that must be used as single ended, all the other can be used as differential lines. Differential lines are a necessity!

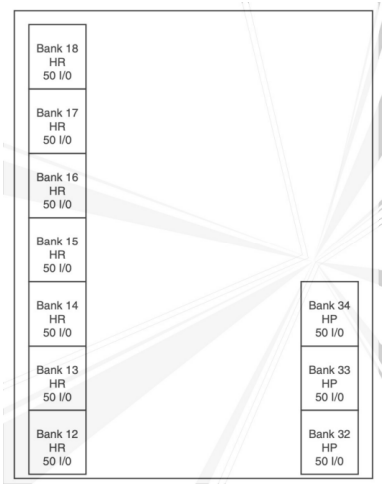


Figura 19: Banks placement

4.2 DCI Digitally Controlled Impedance

«The password (?) of our systems is Signal Integrity»

At the speed we need there are physical phenomena that destroy our signals. The main source of issue is not the speed per se but the fast edges that can cause **reflections** and **ringing** on the lines. The solution is the **termination** of the line.

The issue is not the damaging of the driver due to reflected power, "this is non-sense!", The issue is information loss and spurious signals. The real issue is the **Intersymbolic Interference**, change of a bit due to reflected power.

The DCI allows us to select the internal termination impedance of the trace, such as external components that might introduce non-idealities are not needed. Most common termination techniques are serial and parallel termination. External termination might still be needed in some applications.

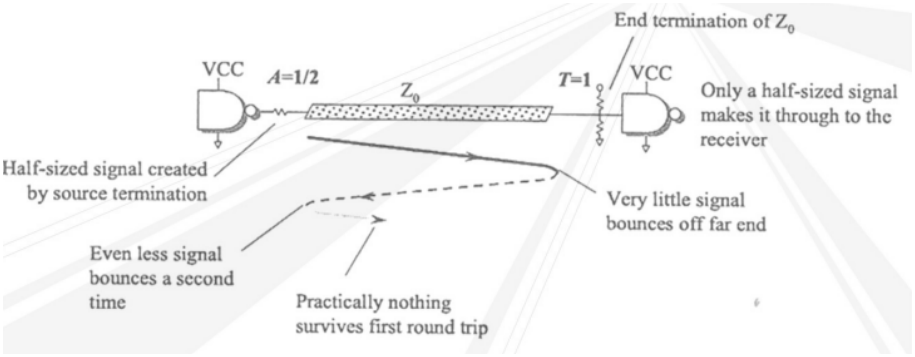


Figura 20: Advanced external termination technique

Bypass capacitors are different values capacitors that are placed in parallel (to the power supply pins) such as the total impedance vs frequency profile (that is a v shaped curve for a single cap due to parasitism) can stay low for a wider range of frequencies.

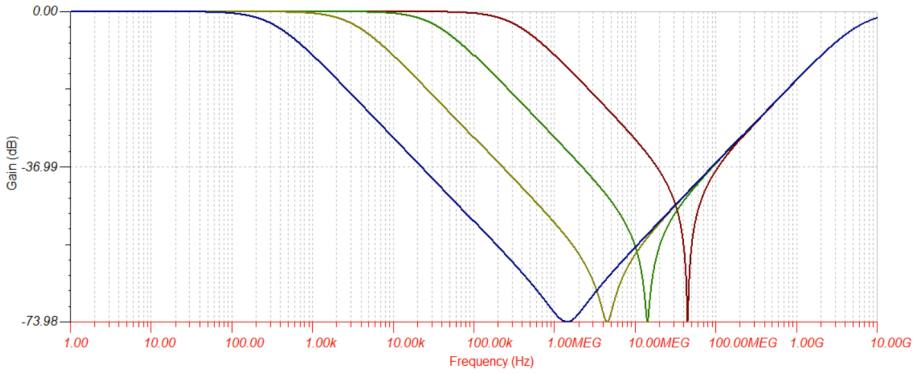


Figura 21: Bypass capacitors

At high frequency the bypass caps shorts the disturbance to ground, but with one cap the bypass is effective only in a very narrow range of frequencies. The capacitor's package also plays an important role in the frequency profile.The lower peaks are at resonance of the LC circuit. The model of a real capacitor is:

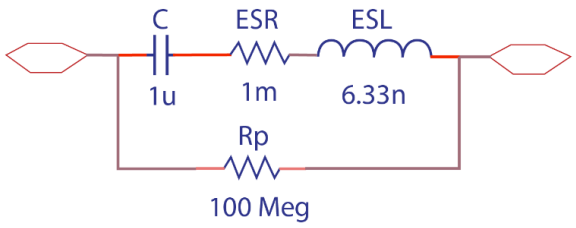


Figura 22: Capacitor real model

Shannon's capacity theorem tells us what's the limit of data-rate in bit/sec that can be sent on a physical channel, knowing the source bandwidth B and received S/N.

$$C = B \log_2(1 + \frac{S}{N}) \tag{1}$$

so in order to increase the data-rate we can only act on noise.

4.3 SelectIO Primitives

The SelectIO drivers and receivers supports many different primitives (not important to know them, just showing):

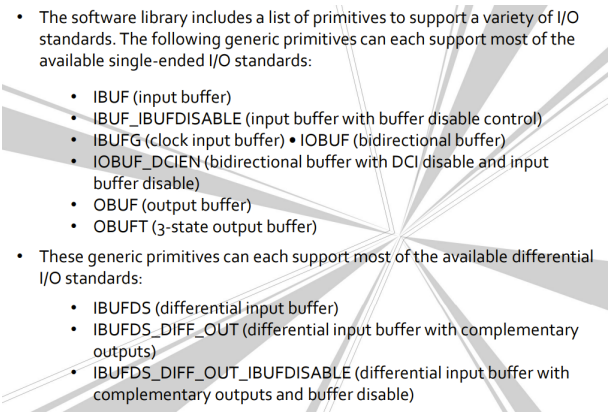


Figura 23: SelectIO primitives

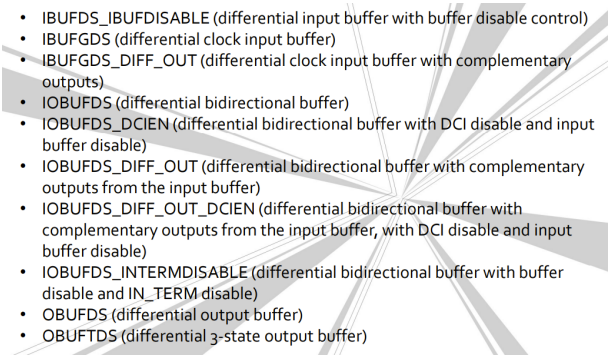


Figura 24: SelectIO primitives

Why are **differential lines** needed? A forward and return path are always present in a circuit, when single ended lines are used the return path is the ground plane beneath the trace in our circuit. The issue is that there are voltage drops since ground is not ideal, and ground for a device might become different from ground of another device making the reference of the two devices not the same.
Differential signal are two balanced voltage signals, meaning they have the same amplitude but opposite polarity, in this way the reference is carried along the signal. The forward and return currents are both carried in the two lines. This is crucial for many standards, like DDR memories. Transmitter and receiver must be capable to handle differential signal.

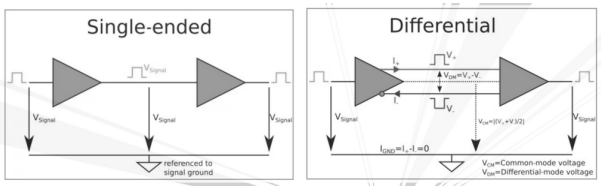


Figura 25: Differential lines vs Single Ended

At high speed, capacitive effect might be caused by everything around our device (even disconnected scopes apparently) so there is current flowing outside our system and we lose information! "As higher the frequency as higher the effect, unavoidable coupling".

«When you can't eliminate an effect, try to compensate it»

The differential lines must be very close and very similar, in such a way that they will **experience the same coupling**, such as by having opposite polarity the same current will leave and enter our system! No net information loss.

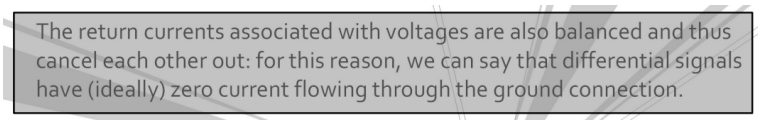


Figura 26: "The Golde Rule"

Also in terms of radiated EMI, the two opposite polarity fields cancel out, greatly reducing the emitted radiation.

4.4 SERDES & IO_FIFO

As seen, the SelectIO drivers and receivers allow to control impedance, slew rate, drive strength (in mA) and pull up/down, but there is further logic.

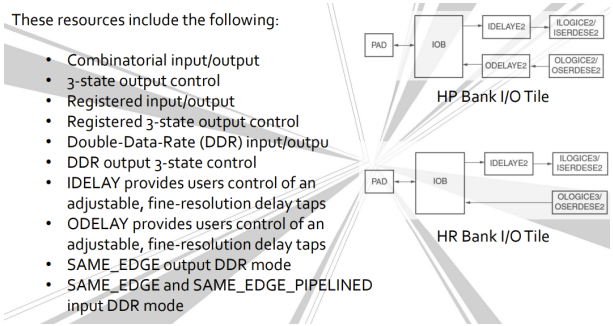


Figura 27: IO Bank logic

An important block is the **SERDES** serial to parallel logic resources, since:

«Communication goes serial, Processing goes parallel»

These are blocks used to take the parallel logic used inside the FPGA and serialize it to transmit it and vice-versa.

IO_FIFO are available in the SelectIO drivers. They are shallower versions of normal FIFOs and are used for clock domain crosses between parallel communication lines.

4.5 Simultaneous Switching Outputs & Simultaneous Switching Noise

Due to **package inductance**, only a limited number of simultaneously switching outputs is supported. The model is the following:

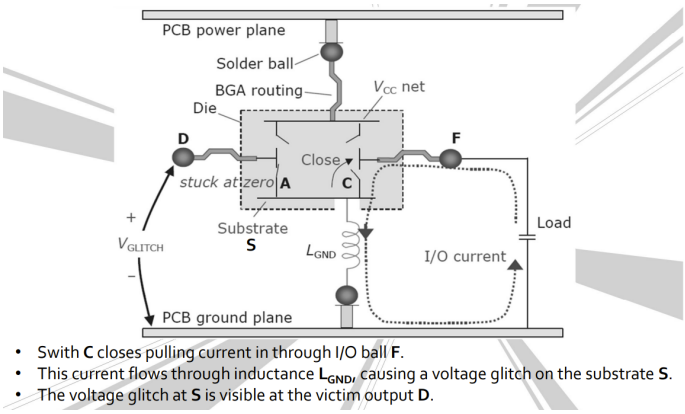


Figura 28: SSO model

At the bottom there is the PCB ground plane. Initially switch A is closed so D sees low level, while switch C is open and the capacitor Cload is charged therefore F sees high logic level. Switch C is closed so Cload is discharged. However there is an **high frequency signal** due to the current step from the discharging C. The inductor L has high impedance at high frequency, there is a voltage drop on the connection between the device and ground! During the drop of voltage the reference of the chip is changed, pin D is not a ground level, but at ground level + the voltage drop, pin D is moving from low level to high level! This can be a great problem since there are thousands of commuting devices in the chip.
Inductance of a pin is not the same for every pin, it depends on the loop that the forward and returning signals go through:

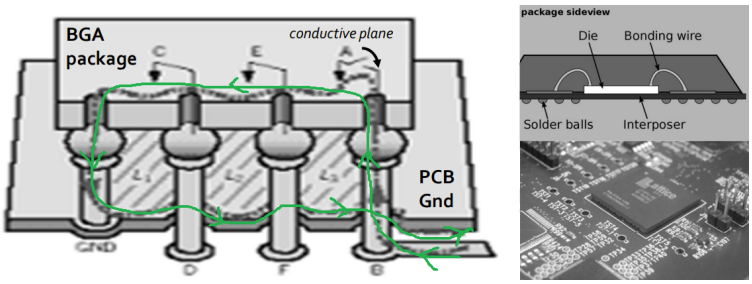


Figura 29: Pin inductance

The circuit model is the following:

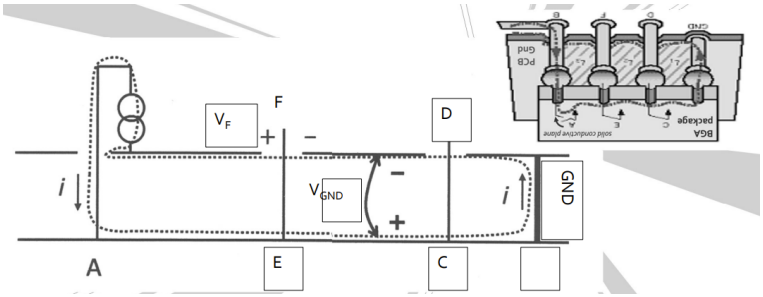


Figura 30: Pin inductance model

where the pins F and D are used as probe to measure the voltage drop between pcb GND and internal GND of the chip. A current is forced inside the circuit.

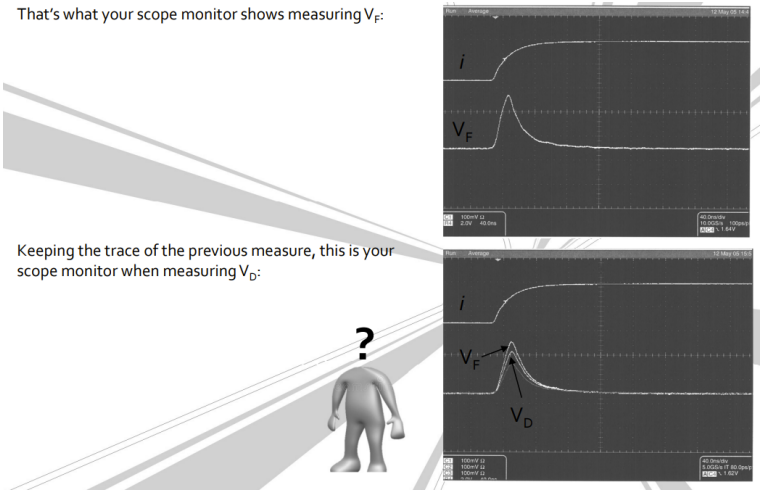


Figura 31: Pin inductance result

Notice how the voltage between GND and D is lower than the drop between GND and F. Measurement is space-dependent, therefore **inductance is a property of the space**, of the topology. This is because:

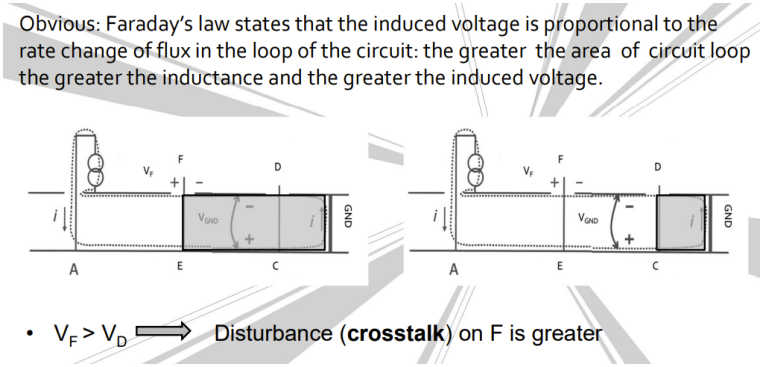


Figura 32: Loop effect

«Golden Rules»

In order to reduce the inductance it's important to:

- choose the **best package**
- increase the number of GND and power pins for smaller loops
- drive smaller loads for smaller di

Also GND must be **distributed** all around the package such as the area of the loops is minimized. Also capacitance is present but it's not activated by the signal.
For cross-talk, if a Faraday cage is built around the signal, nothing changes! The loop between forward path and ground must be reduced in order to reduce it.
A signal has its energy distributed in magnetic and electric field, it's the impedance of the medium that balance the two: below 377Ω there is more concentration of magnetic field, above there is more electric field.

5 Pipeline

«Think parallel»

PLDs are made for parallel computing:

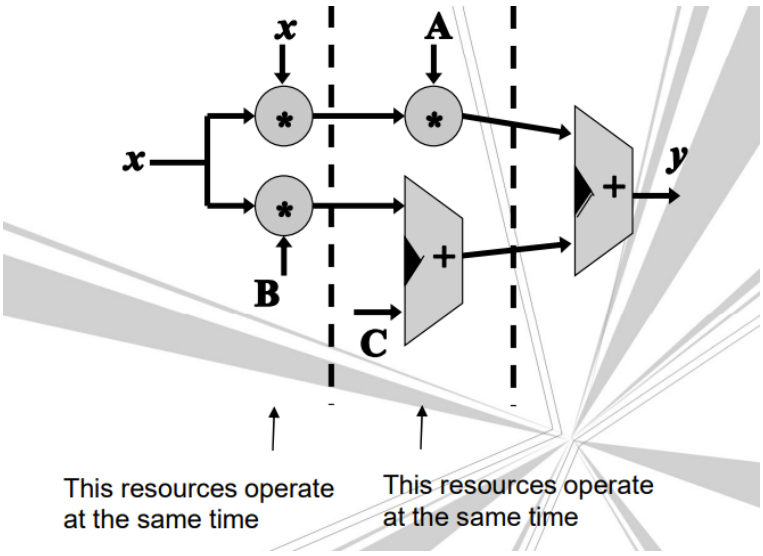


Figura 33: Pipeline

Temporal vs Spatial parallelism:

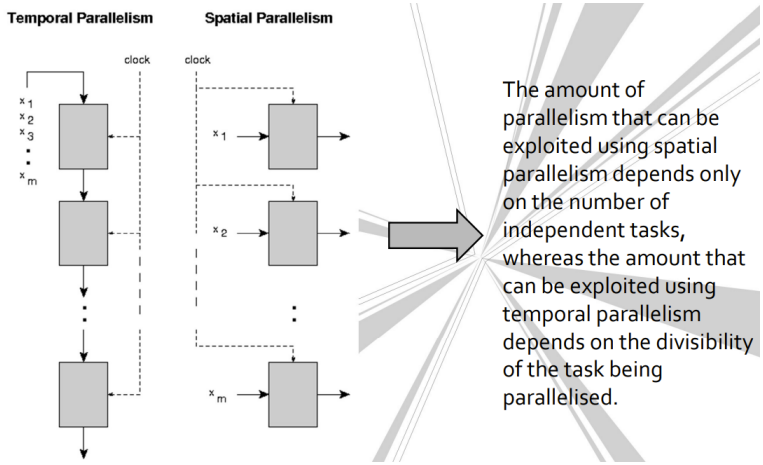


Figura 34: Temporal vs Spatial

5.1 Pipeline structure

To attain pipelining, combinatory logic must be divided in simpler parts separated by registers. Let's start from a non-pipeline structure.

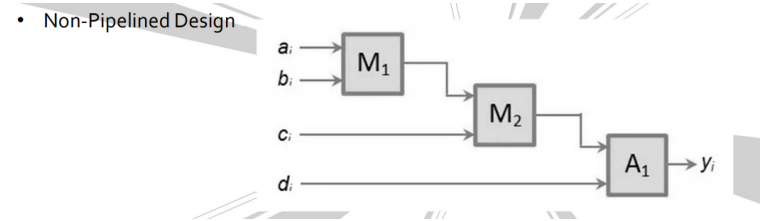


Figura 35: Non pipelined design

In this case between input and output there is the sum of the propagation times along the longest path. There are two parameters in **combinatory circuits**:

- **Latency** τ_{pd}: max delay between input and relative output (its increased with pipeline)
- **Throughput** $1/\tau_{pd}$: frequency of new valid output

Now registers are introduced between combinatory blocks:

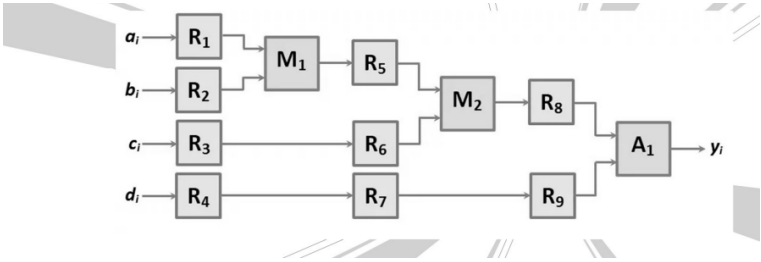


Figura 36: Pipelined design

now the **throughput** is no longer the inverse of latency, but the inverse of the propagation time of the **longest path between two registers**. Remember that registers consumes power tho. Only the latency is increased, but it's only at the beginning so it's not very important.

5.2 Design Methodology

It's crucial that every path between input and output has **the same number of registers**, such as there is coherence between input and output.
There are several steps to make a circuit pipelined:

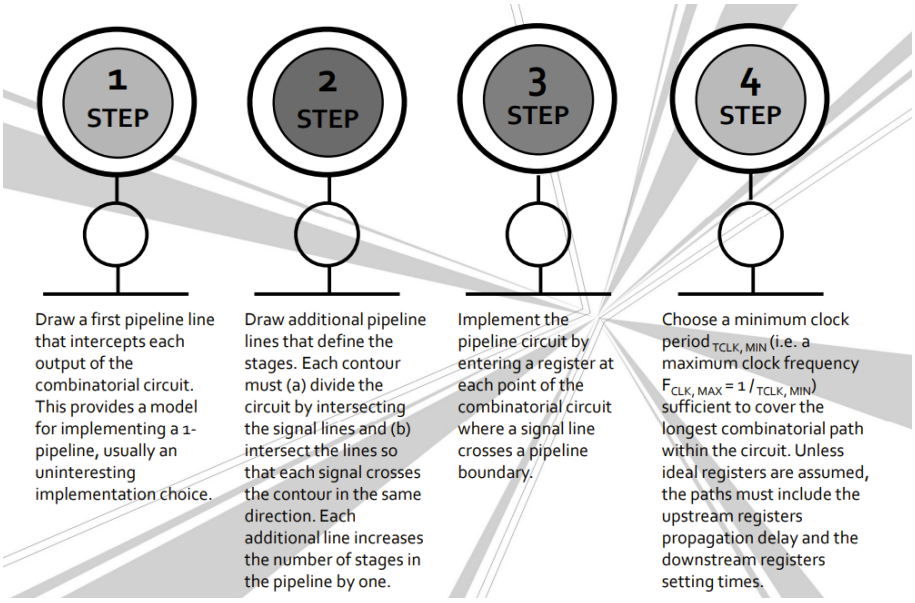


Figura 37: Methodology

here an example:

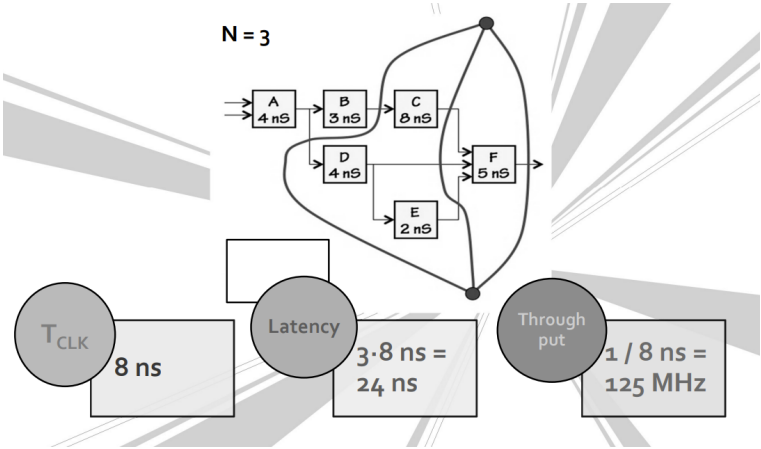


Figura 38: Example

How many steps do we add? When we have isolated the slowest component, we must wait for it, so every further stage we add will only have a negative effect.

5.3 Optimization

Further optimization is possible thanks to **interleaving**:

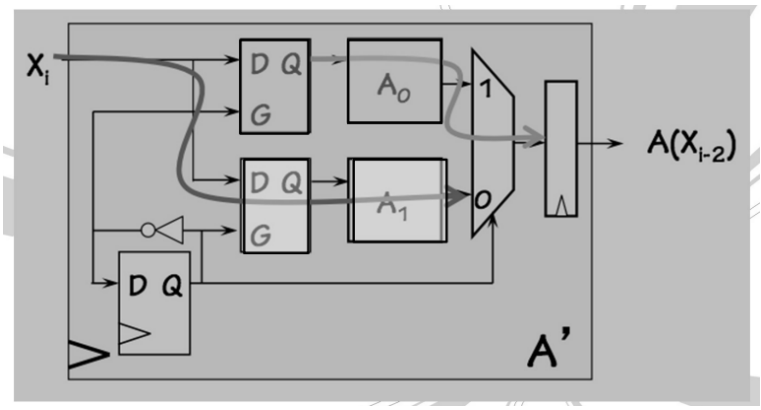


Figura 39: Interleaving structure

This is a method used a lot in ADCs. The idea is to replicate the stage we need (A in the picture) and make them function in parallel. The FF on the bottom is a frequency divider and provides the clock/2 to the other 2 FF. See the signals to understand, I lost my will to live.

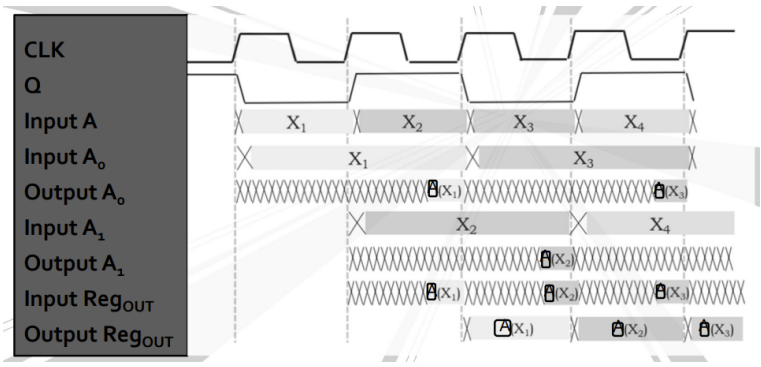


Figura 40: Interleaving signals

«Interleaving is a concept»