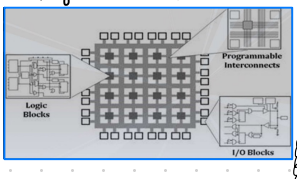
DESD THEORY

1 - FPGA INTRODUCTION

FPGA are made of:
- → CLB (CONFIGURABLE LOGIC BLOCKS)
- → I/O BLOCKS
- → Programmable interconnections



all set with CAD. configurations can change!

↓

REPROGRAMMABILITY

thanks to LOOK-UP TABLES (LUT)
ADDRESSES THE TABLE with INPUTS OF THE FUNCTION for which corresponds a STORED VALUE.
(it's a sort of MEMORY)

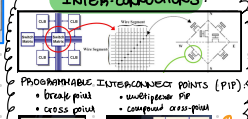
I/O BLOCKS: programmable input or output with
* TRI-STATE BUFFERS for BIDIRECTIONAL CONTROL
* FF/latches
* DELAYED OUTPUT CLOCKS
* PULL-UP/DOWN RESISTORS
* PROGRAMMABLE VOLTAGE and CURRENT LEVELS
* SET BY CONFIGURATION BITS

to transfer data from outside to inside:
- BUFFERS TO DRIVE dedicated I/O lines
- ADD DELAY to I/O
- CLOCK CAPABLE INPUT BUFFER.

GTS: GLOBAL 3-STATE (if HIGH, signal not ABLE I/O to High Impedance)

GSR: GLOBAL SET/RESET (signal set FFs to INITIAL STATE)

INTER-CONNECTIONS:



PROGRAMMABLE INTERCONNECT POINTS (PIP)
- break point
- cross point
- multiplexer PIP
- compound cross-point

THE MORE THE SIGNAL PATH SWITCHES THE HIGHER P.C. LOAD SEEN

POWER

DUE TO SPIKES OF CURRENTS OF ON/OFF of SWITCHES MOS.
↳ complex POWERING SYSTEM, lot of supplies.

⇓

TRANSISTOR BECOMES SMALLER and SMALLER

We can choose:
- SAVE POWER
- SAME POWER, MORE FUNCTIONS

also:

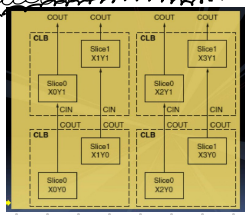
LOW-POWER = RELIABILITY

BLOCK RAM (BRAM)
- about 1Mb (mico-bit) made of smaller RAM blocks independently controlled
- cascade available for wider memory instantiation.
- can be DUAL PORT (W/R)
- in addition to DISTRIBUTED RAM, dedicated blocks.

CLB:

made of
- → LUTs
- → DISTRIBUTED MEMORY
- → SHIFT REGISTERS
- → DEDICATED CARRY LOGIC HIGH-SPEED FOR ARITHMETIC FUNCTIONS
- → WIDE MULTIPLEXERS

each CLB connected to a SWITCH MATRIX for access to general routing matrix.



- A CLB is made of a PAIR OF SLICES (in column)
  - → SLICEL: logic slices
  - → SLICEM: can use LUTs as 64-bit RAM or 32-bit shift registers.

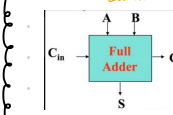
every LUT can use a FF to register the input
slices contains also MULTIPLEXERS and ARITHMETIC CARRY LOGIC.
SLICES not connected between each other.

EVERY SLICE:
- logic LUTs
- storage elements
- wide-function multiplexers
- carry logic
  - → UP TO 5/6 INPUT BOOLEAN FUNCTIONS 'UP TO 7/8' WITH MULTIPLEXER
  - → to perform FAST ADDITION and SUBTRACTION in a SLICE

in SLICEM:
- DISTRIBUTED RAM
- SHIFTING DATA with 32 bit-registers (can be used to realize serial delays)
  - WITH SINGLE or MULTIPLE PORT INTERFACES
  - 1 port: SYNC WRITE (1 clock cycle)
  - 1-3 port: ASYNC READ

CARRY LOOKAHEAD: calculate in advance the carry

THE HIGHER THE INPUT BITS OF ADDITION, THE HIGHER THE CRITICAL DELAY PATH FOR CARRY.
→ SOLUTION: COMPUTE IN PARALLEL ALL POSSIBLE CARRY PATH; AND THEN SELECT THE RIGHT ONE



$C_{OUT} = AB + AC_{IN} + BC_{IN}$
$= AB + (A + B)C_{IN}$
$= G + P\,C_{IN}$
generate propagate

A	B	C_i	S	C_o	Carry status
0	0	0	0	0	delete
0	0	1	1	0	delete
0	1	0	1	0	propagate
0	1	1	0	1	propagate
1	0	0	1	0	propagate
1	0	1	0	1	propagate
1	1	0	0	1	generate
1	1	1	1	1	generate

$C_{0,0} = P_0 C_{IN} + G_0$
$C_{0,1} = P_1 P_0 C_{IN} + P_1 G_0 + G_1$
$C_{0,2} = P_2 P_1 P_0 C_{IN} + P_2 P_1 G_0 + P_2 G_1 + G_2$
$C_{0,3} = P_3 P_2 P_1 P_0 C_{IN} + P_3 P_2 P_1 G_0 + P_3 P_2 G_1 + P_3 G_2 + G_3$
→ can be referred to first inputs

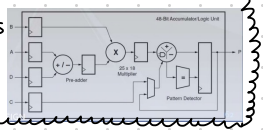
STACKED SILICON INTERCONNECT: to provide connections between SUPER-LONG LINES (SLL) and between SUPER LOGIC REGIONS (SLRs)
↳ SUPER-IMPOSED DIE OF SILICON
⇒ PUSH PERFORMANCES

PACKAGE

Very important for INDUCTANCE EFFECTS, to be minimized with OPTIMAL PLACEMENT AND DISTRIBUTION OF POWER and GND PINS.
⇒ INDUCTANCE DESTROY SIGNAL

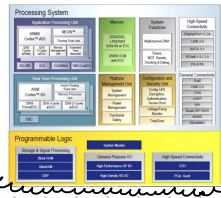
DSP: DIGITAL SIGNAL PROCESSING

can implement CUSTOM PARALLEL ALGORITHMS up to 25x18 CPLX, 48 bit-ACCUMULATOR, POWER SAVING PRE-ADDER, ARITHMETIC UNIT, LOGICAL UNIT, PATTERN DETECTOR



SYSTEM ON CHIP (SoC)

PROCESSOR SYSTEM + PROGRAMMABLE LOGIC

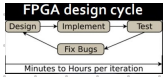


FPGA vs ASIC (Application Specific IC)

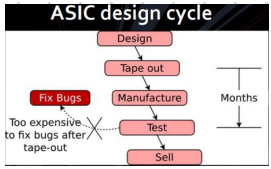
FPGA are re-configurable (also while running)
ASIC are permanent, cannot be changed

- ASICs SUITABLE FOR HIGH VOLUME PRODUCTIONS
- FPGA for PROTOTYPING and VALIDATING DESIGN (also of ASIC designs)
- FPGA LIMITED IN FREQUENCY compared to ASIC (due to ROUTING and BLOCKS)
  ASIC with same FPGA are much FASTER since they are optimized for specific function.
- DESIGN OPTIMIZATION ARCHITECTURE can SPEED-UP A LITTLE BIT the FPGA
- ASICs are more POWER EFFICIENT
- ANALOG DESIGN ARE NOT POSSIBLE WITH FPGA (but today they can incorporate some analog stages like ADC and PLLs, however not so much flexibee to create compeex circuits like a RF transceiver)
  ↳ ASICS can have complete analog design (or mixed)
- FIELD PROGRAMMABLE ANALOG ARRAY (FPAA) failed due to too NARROW BAND LIMITS due to SWITCH MATRIX ARCHITECTURE.

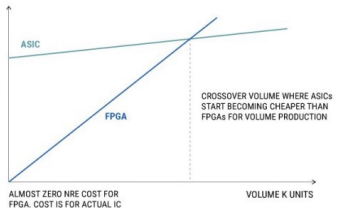
→ USED FOR PROTOTYPING
NO NRE, FREE SOFTWARE
ONLY IC COST of BOARD



HIGH NON RECURRING COST (NRE)
(IC MASK IN FOUNDRY)
DIFFICULT FOR PROTOTYPING
(COSTLY IN SMALL QUANTITIES)

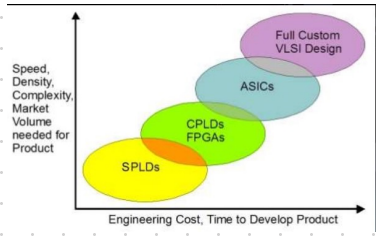


ASIC cost components:
- ASIC EDA tools and training
- Cost of designing
- DFT cost
- Cost of simulating
- ASIC Masks Cost
- Wafer Cost
- Wafer Processing
- Die Utilization
- Yield & Manufacturing Loss
- Packaging



- RESET TREE
- CLOCK TREE
- LAYOUT
- MANUFACTURING
- TESTING

BOTH DESIGNED with VHDL
ASIC REQUIRE MUCH MORE ATTENTION IN IMPLEMENTATION, which is done almost automatically in FPGA CAD



FPGA vs ASIC
Visual comparison

	FPGA	ASIC
NRE	✓	
Performance		✓
Time to market	✓	
Design Flow	✓	
Cost per Unit (high volume)		✓
Barrier to Entry	✓	
Energy Efficiency		✓
Analog Blocks		✓

NUMATO LAB

DESIGNING in FPGA : exploit PARALLELISM!

Temporal Computation

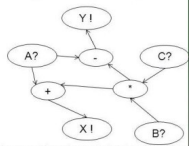
The traditional paradigm
Typical of Programmers
Things done over time steps

```
A = input("A= ? ");
B = input("B =? ");
C = input("B multiplier ?");
X = A + B * C
Y = A – B * C
```

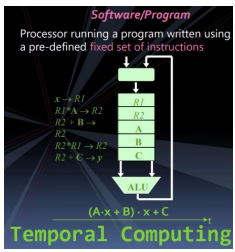
Which do you think is easier to make sense of?

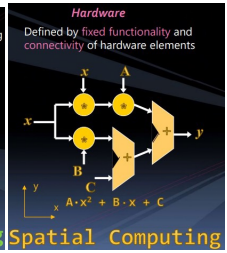
Spatial Computation

Suited to hardware
Possibly more intuitive?
Things related in a space



Can provide a clearer indication of relative dependencies.

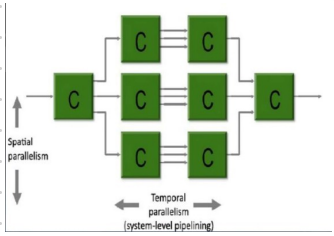


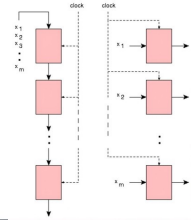


TEMPORAL PARALLELISM:
require PARTITIONING of a PROCESSING TASK
→ PIPELINE STRUCTURES
LATENCY ↑
THROUGHPUT ↑
→ depends on DIVISIBILITY OF TASK to be parallelized

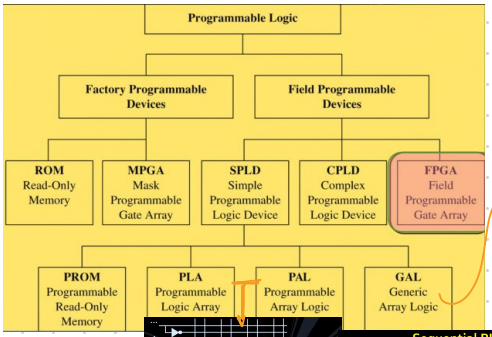
SPATIAL PARALLELISM:
replicate the processing task to be computed by its own dedicated component
→ units of INPUT INFORMATION must be PARTITIONABLE.
→ depends on NUMBER of INDEPENDENT TASKS

ADVANTAGES:
- RE-PROGRAMMABILITY
- PERFORMANCES SUFFICIENT FOR MOST APPLICATIONS
- MASSIVE PARALLELISM
- I/O FLEXIBILITY to interface to MEMORY
- BUGS RISK REDUCED
- FAST TIME TO MARKET
- REASONABLE POWER CONSUMPTION and THERMAL BEHAVIOR
- LOW BILL OF MATERIALS (BOM COSTS)

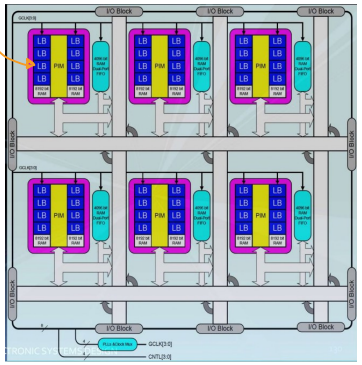


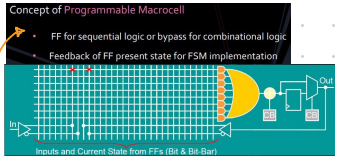


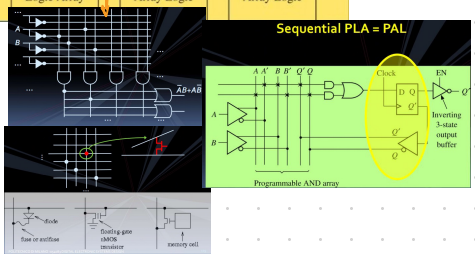
PROGRAMMABLE LOGIC DEVICE (PLDs)



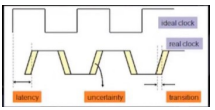
CPLD → ARRAY OF PLD

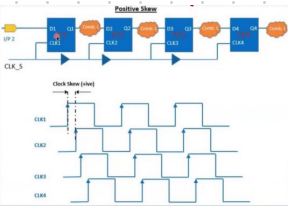






2 - CLOCK AND TIMING

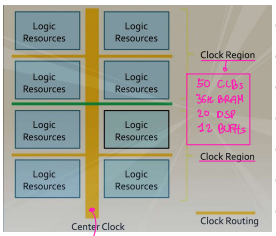




→ THERE ARE SOME NON-IDEALITIES IN REAL CLOCKS
- JITTER (unavoidable, always present)
- SKEW (due to delays of the lines) → GOOD CLOCK ARCHITECTURE NEEDED

→ FPGAs has DEDICATED GLOBAL and REGIONAL I/O and CLOCKING RESOURCES
↳ CLOCK MANAGEMENT TILES (CMT) provide: * clock freq. synthesis * deskew * jitter filtering ⇒ LOCAL ROUTING not reccomended

⇒ MULTIPLE CLOCK REGIONS supplied by CLOCK TREES:



- each CLB is connected to the GLOBAL ROUTING NETWORK through SWITCH MATRIX
- DEDICATED LINES for CLOCK PROPAGATION, to avoid SKEW.
  DRIVEN BY BUFG (GLOBAL CLOCK BUFFER) → drive every region through HROW
- each I/O BANK contains CLOCK-CAPABLE INPUT PINS
- HORIZONTAL CLOCK BUFFER (BUFH) allows access to CLOCK LINES of a SINGLE CLOCK REGION
- MULTI-CLOCK REGION BUFFERS (BUFMR) to span up to 3 vertically adjacent clock regions
- BUFIO for I/O CLOCK TREE
- BUFR (REGIONAL CLOCK BUFFER)

VERTICAL CLOCKING CENTER LINE
BACKBONE

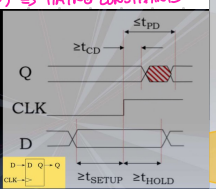
CMT
- → MMCM: mixed-mode clock manager module
- → PLL: phase-locked loop

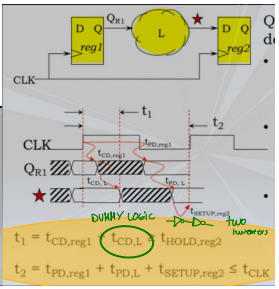
for FREQs synthesis, deskew, jitter filtering

HIGH PERFORMANCE CLOCK ROUTING with
-) low jitter
-) minimal duty cycle distorted direct path

→ STILL THERE ARE DELAYS (deterministic or stocastic) ⇒ TIMING CONSTRAINTS

- t_{PD}: maximum propagation delay CLK → Q
- t_{CD}: minimum contamination delay CLK → Q
- t_{SETUP}: minimum time D stable before clock edge.
- t_{HOLD}: minimum time D stable after clock edge.





3- I/O RESOURCES

FPGAs offers both:

a) HIGH-PERFORMANCE (HP) ⇒ for HIGH SPEED MEMORIES

b) HIGH-RANGE (HR) ⇒ to support WIDE RANGE of I/O standards (with different voltages)

↳ every I/O pin contains INPUT, OUTPUT, 3-STATE DRIVERS

⇒
- PROGRAMMABLE OUTPUT STRENGHT and SLEW-RATE, ON-CHIP TERMINATION with digital-controlled impedance
- internally generated REFERENCE VOLTAGE
- configurable SelectIO PINS, both SINGLE-ENDED or DIFFERENTIAL (2 pins)

⇓

HIGH PERFORMANCE FAST SYSTEM ⇒ DIFFICULT MANUFACTURING & DESIGN (WORSE SIGNAL INTEGRITY)

due to REFLECTIONS and RINGING which require proper LINE TERMINATION

- NO NEED OF TERMINATION RESISTOR ON-BOARD
- REDUCE BOARD ROUTING DIFFICULTIES
- IMPROVE SIGNAL INTEGRITY BY ELIMINATING STUB REFLECTION

↳ RESISTORS placed in the LINE to match IMPEDANCE of DRIVER/RECEIVER

DCI used because a real resistor can be impossible to place near pins, or create stub effects.

(DIGITALLY CONTROLLED DCI) IMPEDANCE

↳ CAN CONTROL: - OUTPUT Z OF A DRIVER - ADD PARALLEL TERMINATION TO MATCH Z OF TRANSMISSION LINE

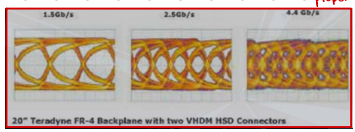
ADAPT to PROCESS & TEMP VARIATIONS

if resistor too far from end of the line

SHANNON theory

$$C = B\log_2\left[1 + \frac{S}{N}\right]$$

C: CAPACITY in BIT/S — B: BANDWIDTH — S/N: SIGNAL to NOISE RATIO

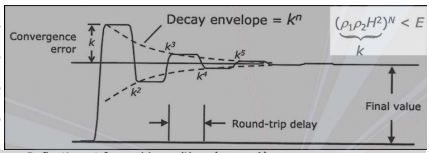


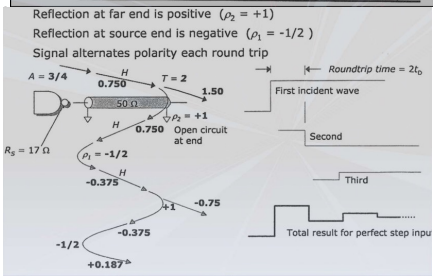
due to - CROSSTALK BETWEEN CONNECTIONS
- REFLECTIONS

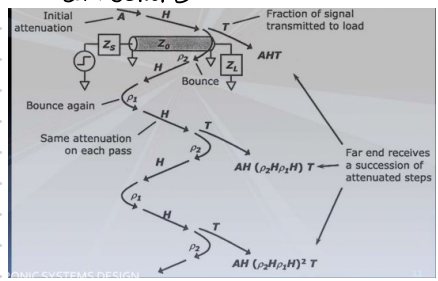
$$\rho = \frac{Z_{term} - Z_0}{Z_{term} + Z_0}$$

REFL. COEFF.

difficult to achieve $Z_{term} = Z_0$ because HIGH INPUT IMPEDANCE of MOS ≠ IMP. OF LINE

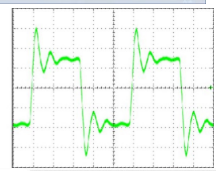






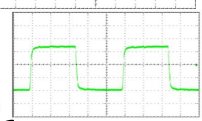
Ringing

- Source impedance too low
- Load impedance too high
- Initial step too big (overshoot)



Stepwise

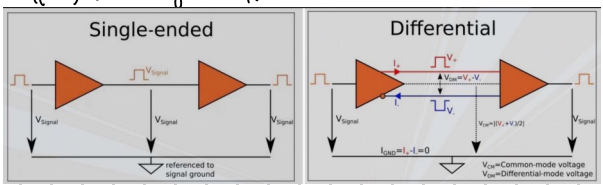
- Source impedance too high
- Load impedance too high
- Initial step too small (doesn't meet VOH)



HOW TO SOLVE:

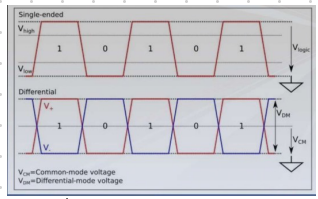
- End termination
- Series termination ← Most popular styles for digital logic
- Both ends termination
- Any of these approaches can prevent problems with long-line reflections or short-line ringing

↳ LIBRARY support total freedom on I/O output/input/bidirectional buffers, both single or differential.



DIFFERENTIAL is better: RETURN CURRENTS BALANCED and CANCEL EACH OTHER OUT (ideally zero current flowing through the ground connection)

also reduced effect for external EMI and CROSSTALK and also emitted ones. (contributions are equally added to inverted and not-inverted signal)



- LESS POWER FOR SAME SNR
- BETTER NOISE MARGINS (HIGH or LOW STATE DETERMINATED BY COMPARING NON-INVERTED and INVERTED SIGNAL)

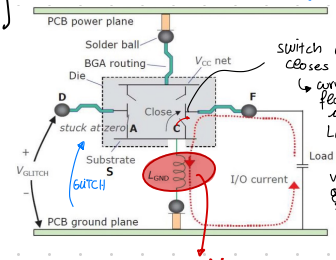
↳ CROSS-OVER POINT, SHIFTS IF TWO WIRES HAVE DIFFERENT LENGTH

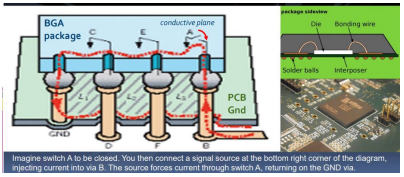
- There is a TRADE-OFF between POWER/PERFORMANCE (delay of input buffers). We can choose that DELAY.
- We can also choose a SLEW-RATE but HIGHER means also MORE REFLECTIONS (and NOISE).
  STRENGHT (in mA)
  PULL-UP/DOWN or KEEPER CIRCUIT

↳ **COMMUNICATION STANDARDS**: COMMUNICATION GOES SERIAL, PROCESSING GOES PARALLEL

→ SERIAL-to-PARALLEL CONVERTERS present
→ PARALLEL-to-SERIAL " present
→ IO FIFOS

⇒ due to PACKAGE INDUCTANCE, each package supports a limited number of SIMULTANEOS SWITCHING OUTPUTS (SSOs) (PARTICULARLY WHEN HIGH-DRIVE OUTPUTS ARE USED HIGH)



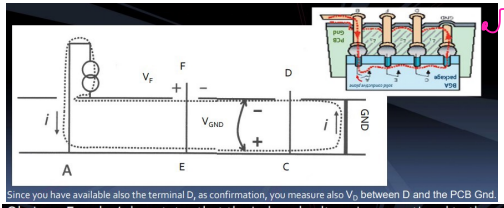


PARASITIC INDUCTANCE

NOT ALL PINS are EQUAL, BECAUSE INDUCTANCE IS NOT A PROPERTY OF ANY INDIVIDUAL WIRE BUT IT'S A PROPERTY OF THE SPACE BETWEEN CONDUCTORS

MECHANISM OF COUPLING AMONG PINS GENERATING INTERFERENCES that is called

SIMULTANEOUS SWITCHING NOISE (SSN)

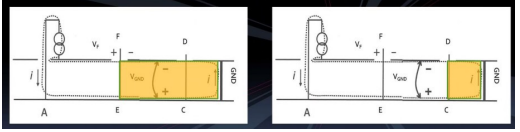


→ we can MEASURE L_{GND} as

$$V_{GND} = L_{GND} \cdot \frac{di}{dt}$$

but we measure V_F between F and GND (since we do not have access to the conductive plane inside the package)

Obvious: Faraday's law states that the induced voltage is proportional to the rate change of flux in the loop of the circuit: the greater the area of circuit loop the greater the inductance and the greater the induced voltage.



$L_{GND}|_{V_F} > L_{GND}|_{V_D}$

$V_F > V_D$ DISTURBANCE (CROSSTALK) ON F is GREATER.

can be reduced:

- REDUCING PIN INDUCTANCE
  - use better packaging with reduced lead inductance
  - increase the number N of power and ground pins
  - drive fewer loads or smaller loads
- DISTRIBUTE UNIFORMLY GROUND PINS TO REDUCE RETURN CURRENT LOOPS

↳ there are software to predict SSN and NOISE MARGINS selecting a VICTIM PIN (considering all AGGRESSORS PINS) (over 8mA)

→ PLACE STRONG OUTPUTS and/or SSOs SEPARATED FROM SENSITIVE INPUTS/OUTPUTS (HIGH-SPEED)

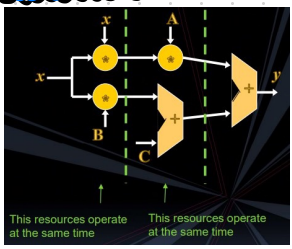
→ ADD VIRTUAL GROUNDS to reduce SSO

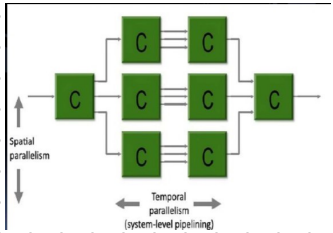
4 - PIPELINE

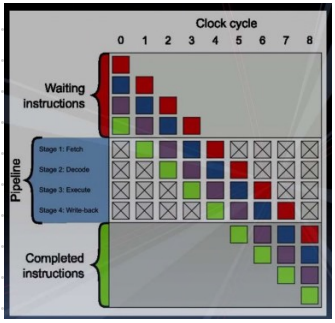
FPGA is made of a sea of CLBs that can operate at the same time

⇒ PARALLEL PROCESSING

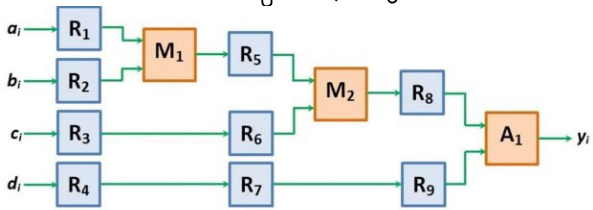
- TEMPORAL PARALLELISM = PARTITIONING OF TASK = PIPELINE ⇓ LATENCY OF SINGLE RESULT ↑ but also THROUGHPUT ↑
- SPATIAL PARALLELISM: replicating processing task if original information can be partitioned







↳ in FPGAs PIPELINE is achieved by interposing REGISTERS between blocks



→ we call LATENCY the DELAY associated with the number of clock cycles lost before the first valid output appears.
→ the greater the number of pipeline stages the greater the latency

→ THE LONGEST PATH (in DELAY) BETWEEN REGISTERS sets the f_{CLK}^{MAX}

→ the THROUGHPUT is much higher (1 output per clock cycle)

⇒ we have to wait initially then great advantage with throughput.

- each path should have the same N number of registers

HOW TO PIPELINE

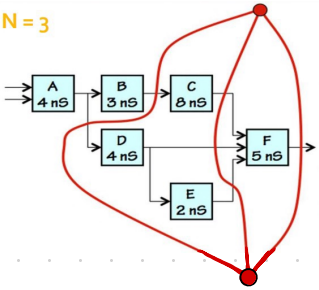
1/ DRAW A 1ST PIPELINE that INTERCEPTS EACH OUTPUT OF COMBINATORIAL CIRCUIT

2/ DRAW ADDITIONAL PIPELINE LINES THAT DEFINE THE STAGES
a) intersect signal lines
b) each signal lines crosses the contour in the same direction

3/ INSERT THE REGISTERS at each INTERCEPTION

4/ CHOOSE A MINIMUM CLOCK PERIOD $T_{CLK,MIN}$ SUFFICIENT TO COVER THE LONGEST COMBINATORIAL PATH WITHIN THE CIRCUIT. (must include $T_{DELAY\ REG}^{1ST}$, $T_{SETUP\ REG}^{2ND}$)

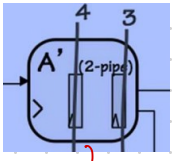
LATENCY: $N \cdot T_{CLK}$
THROUGHPUT: $1/T_{CLK}$



⇒ T_{PD} and T_{SETUP} of FF
↳ WORSENING PERFORMANCE

HOW TO SPEED-UP PERFORMANCE EVEN MORE? HOW TO ELIMINATE THE BOTTLENECK OF THE SLOWEST PATH?

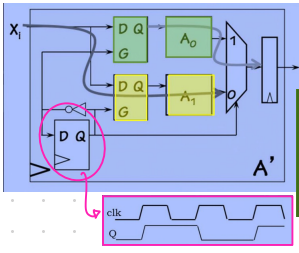
1. PIPELINE INSIDE THE LIMITING LOGICAL BLOCK

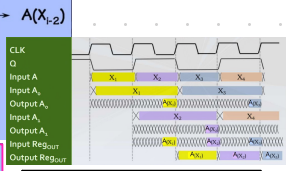


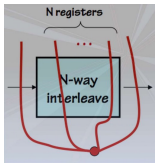
- DIFFICULT TO MODIFY AN IP

2. INTERLEAVING: AREA ↑ f_{MAX} UPPER LIMIT SET BY FF OF PIPELINE

Use the same logical block two times in sync, alternating them with a MUX







INTERLEAVING with N REPLICAS = N-stage PIPELINE

5 - BUSES

a BUS is a COMMON ELECTRICAL PATHWAY BETWEEN MULTIPLE DEVICES, both:
- INTERNAL: ex. data from/to the ALU
- EXTERNAL: ex. for CPU-memory or I/O device connection.

- must obey some COMMON RULES that BUS have to foreow (in designing) **BUS PROTOCOL**
- there are also **MECHANICAL/ELECTRICAL SPECS**

•) devices that can initiate bus transfers are called **MASTERS**. → most connected to a **BUS DRIVER** (digital amplifier)

•) devices that are passive and wait for request are called **SLAVES**. → most connected to a **BUS RECEIVER**.

There are

SYNCHRONOUS BUS:
- has a fixed clock driven by a REFERENCE OSCILLATOR Square-wave like.
- every operation take an integer number of cycles (BUS CYCLES)

☹ WE COULD WASTE SOME TIME: 3.3 cycles → 4 cycles
DIFFICULT TO IMPROVE ONCE BUS CYCLE IS FIXED

☺ IMMUNITY to CROSSTALK
happens at switches we can wait to sample until cross-talk noise settle

ASYNCHRONOUS BUS: DOES NOT HAVE A CLOCK.
- bus cycles can be of any length and not the same.
- handeed by MIXED TECHNOLOGY

☺ NO CLOCK SKEW
LOW POWER
NO NEED TO RESPECT A CERTAIN UNIT OF SLOW PATH
NO NEED OF OPTIMIZATION OF CLOCK (VS TEMP/FABRICATION)
AUTOMATIC ADAPTIVE SPEED

☹ DIFFICULT TO DESIGN
HAZARDS AND GLITCHES
SENSITIVE TO CROSSTALK ALL THE TIME.

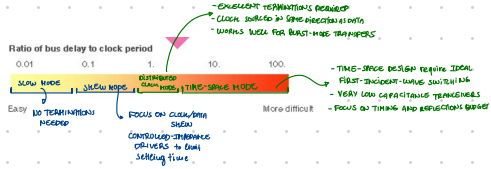
TO GET MORE DATA OVER THE BUS:
- MORE DATA LINES (difficult to obtain identical electrical characteristics)
- FASTER BUS SPEEDS (limited by SKEW)

→ DATA TRAVERS THE BUS AT DIFFERENT SPEEDS

HOW HARD IS TO DESIGN A BUS ?

$$\frac{T_{DELAY}}{T_{CLOCK}}$$ (of length of wires)

figure of merit of difficulty of bus design



NUMBER OF LINES OF BUS = BUS WIDTH ⇒ BANDWIDTH → larger buses are more expensive

to INCREASE: - decrease BUS CYCLE TIME
- increase WIDTH

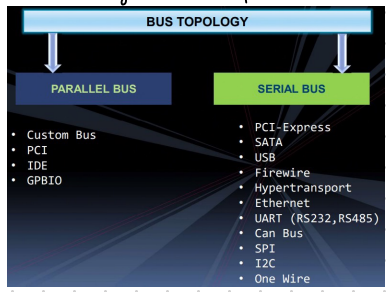
Speeding-up results in problem of BUS SKEW of individual lines travels at slightly different speeds → ☹ BUS NOT COMPATIBLE WITH PRE-EXISTING DEVICES

another solution is to use multiplexed lines for addressing and data transfer → COMMUNICATION IN MULTIPLE STEPS IS SLOW

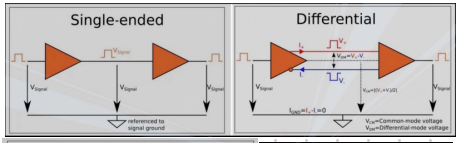
A BUS can be

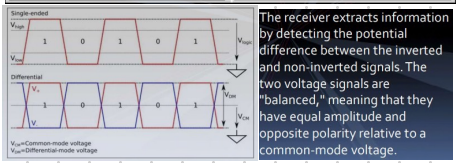
SERIAL:
nowadays it's the preferred choiche for high-speed buses
NO SKEW PROBLEMS (line-to-line signal)
MORE FLEXIBLE APPROACH TO DATA RATES (allows longer cables)
REDUCTION OF LINES (SAVE BOARD SPACE)
FASTER WITH HIGH-CLOCKS
FULL-DUPLEX (specific wires for TX, RX)

PARALLEL: suffers of CROSS-TALK, SKEW, PIN COUNT
NO DELAY for DECODING
SAME STRUCTURE BOTH FOR RX and TX (HALF-DUPLEX)
current in wire ⇒ EM field affect near wires
each bit of data is a wire + other wires for control signal
IMPOSSIBLE TO CREATE IDENTICAL WIRES
RIGHT ORDER OF ARRIVAL OF DATA IS LOST



DIFFERENTIAL SIGNALING:



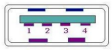


The receiver extracts information by detecting the potential difference between the inverted and non-inverted signals. The two voltage signals are "balanced," meaning that they have equal amplitude and opposite polarity relative to a common-mode voltage.

→ same parasitics, opposite current, NO OVERALL PARASITIC EFFECT
MORE WIRES (and AREA, POWER CONSUMPTION)
preferred for >MHz communications
ideally zero current flowing towards GND
TOLERANCE to external EM interference
LOWER VOLTAGE OPERATION
MORE IMMUNITY TO NOISE, BIGGEST STATIC NOISE MARGIN

BUS TECHNOLOGIES :

USB :



Type A socket (From Front)

Type B socket (From front)

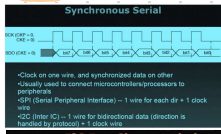
Pin No.	Signal	Color of the cable
1	+5V power	Red
2	- Data	White / Yellow
3	+Data	Green / Blue
4	Ground	Black/Brown

USB SOCKETS & PINS

	Version	First available	Backward compatible	Speed	Bits/sec	HD movie 25 GB transfer
USB	USB 1.1	1998	-	Low speed (LS) Full speed (FS)	1,5 Mbps 12 Mbps	~9,25 h
USB	USB 2.0	2001	USB 1.1	High Speed (HS)	480 Mpbs	~14 min
USB	USB 3.0	2009	USB 1.1/2.0	SuperSpeed (SS)	5 Gbps	~70 sec
USB	USB 3.1	2014	USB 1.1/2.0/3.0	SuperSpeedPlus (SSP)	10 Gbps	~35 sec

- PC host controller HW and SW
- Robust connectors and cable assemblies
- Peripheral friendly master-slave protocols
- Expandable through multi-port hubs
- Low cost
- Auto-configuration
- Hot plugging
- Outstanding performance
- Powered bus

I²C :



- Clock on one wire, and synchronized data on other
- Usually used to connect microcontrollers/processors to peripherals
- SPI (Serial Peripheral Interface) – 1 wire for each dir + 1 clock wire
- I2C (Inter IC) – 1 wire for bidirectional data (direction is handled by protocol) + 1 clock wire

I²C Bus Characteristics

- Includes electrical and timing specifications, and an associated bus protocol
- Two wire serial data & control bus implemented with the serial data (SDA) and clock (SCL) lines
  - For reliable operation, a third line is required: Common ground
- Unique start and stop condition
- Slave selection protocol uses a 7-Bit slave address
  - The bus specification allows an extension to 10 bits
- Bi-directional data transfer
- Acknowledgement after each transferred byte
- No fixed length of transfer
- True multi-master capability
  - Clock synchronization
  - Arbitration procedure
- Transmission speeds up to 100Khz (classic I2C)
- Max. line capacitance of 400pF, approximately 4 meters (12 feet)
- Compatible with different IC technologies

- **Master:**
  - Initiates a transfer by generating start and stop conditions
  - Generates the clock
  - Transmits the slave address
  - Determines data transfer direction
- **Slave:**
  - Responds only when addressed
  - Timing is controlled by the clock line

SPI (Serial Peripheral Interface) :

- As compared with its counterpart I²C, SPI is more suited for data stream applications.

  Communication between DSPs.
  ADC
- SPI can also achieve significantly higher data rates than I²C.
- Full Duplex capability.

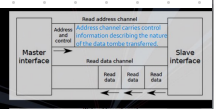
-SPI defines four types of signals.

SPI Signal	Name
MISO	Master In Slave Out
MOSI	Master Out Salve In
SCLK	Serial Clock
SS	Slave Select

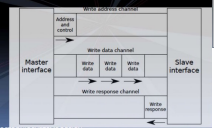
AXI :

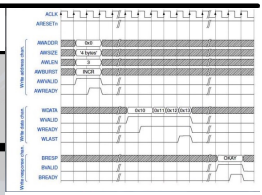
- read address,
- read data,
- write address,
- write data,
- write response.

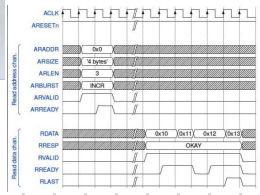
- Read transaction
  - Read data channel transfers data from the slave to the master.



- Write transaction
  - Write data channel transfers data from the master to the slave. In a write transaction, the slave uses the write response channel to signal the completion of the transfer to the master.

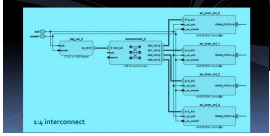






N:1 interconnect. An N:1 interconnect handles the access of a single slave from N masters.
This is managed by an arbiter block: when more than one master tries to issue an operation by raising xVALID in the same clock cycle, the interconnect keeps the corresponding xREADY signals low for every master except for one, whose request is passed unmodified to the slave (if the address is correct).
The priority of the masters is usually configurable, ranging from a round-robin arbitration to a fixed-priority one.

1:M interconnect. On the other hand, a 1:M interconnect connects a single master to M slaves. Each slave has a (usually static) address range, composed by an offset and a range.
When the master performs a read/write operation, the interconnect looks up if the specified address is mapped by one of the slaves: if so, it forwards the operation to the correct slave.


1:4 interconnect

- **N:M interconnect.** Once 1:N and N:1 interconnects have been defined, it is possible to construct an N:M interconnect by combining them.
  The arbiter and router blocks can be connected either in full crossbar mode (two masters can access to two different slaves during the same clock cycles), which leads to higher performance but uses more resources, or in shared access mode, where only one device can access the channel during each clock cycle.
  Usually, the address channel bandwidth requirements are much lower than the data ones (each read/write address can lead to up to 256 data transfers) so, in general, a SAMD (Shared-Access Multiple-Data is preferred.

When in SAMD mode, the N-to-M use case of the AXI Interconnect core provides for only one transaction at a time. For each connected master, read transaction requests always take priority over writes. The arbiter then selects from among the requesting masters. A write or read data transfer is enabled to the targeted slave device. After the data transfer (including the write response) completes, the next request is arbitrated. SAMD mode minimizes the resources used to implement the module of interconnection.

There also exists a simplified subset of AXI4, called AXI4-Lite, whose key limitations are:

- all transactions are of burst length 1,
- all data accesses use the full width of the data channel,
- all accesses are Non-modifiable, Non-bufferable,
- Exclusive accesses are not supported.

Basically, AXI4-Lite retains the structure of AXI4 but removes most of the auxiliary signals and supports only 1-word transactions.

AXI4-Stream is another subset of AXI4 described in the AMBA specification. Unlike AXI4 and AXI4-Lite, in AXI4-Stream the data flows only in one direction, from the master to the slave, and no address is transmitted.

The signals employed are just:

- ACLK and ARESETn,
- TDATA, TVALID and TREADY for the basic handshake,
- TLAST to mark the end of the stream,
- TSTRB and TKEEP to mark a byte as data, null or positional,
- TID to identify different streams,
- TDEST to provide routing informations,
- TUSER for a user-defined sideband signal.

The handshake mechanism in AXI4-Stream is the same used in AXI4 and AXI4-Lite.