PROGETTAZIONE DI SISTEMI DIGITALI

[Fotocopie di Appunti PARTE 2]

A CURA DI ALESSANDRO PAGHI

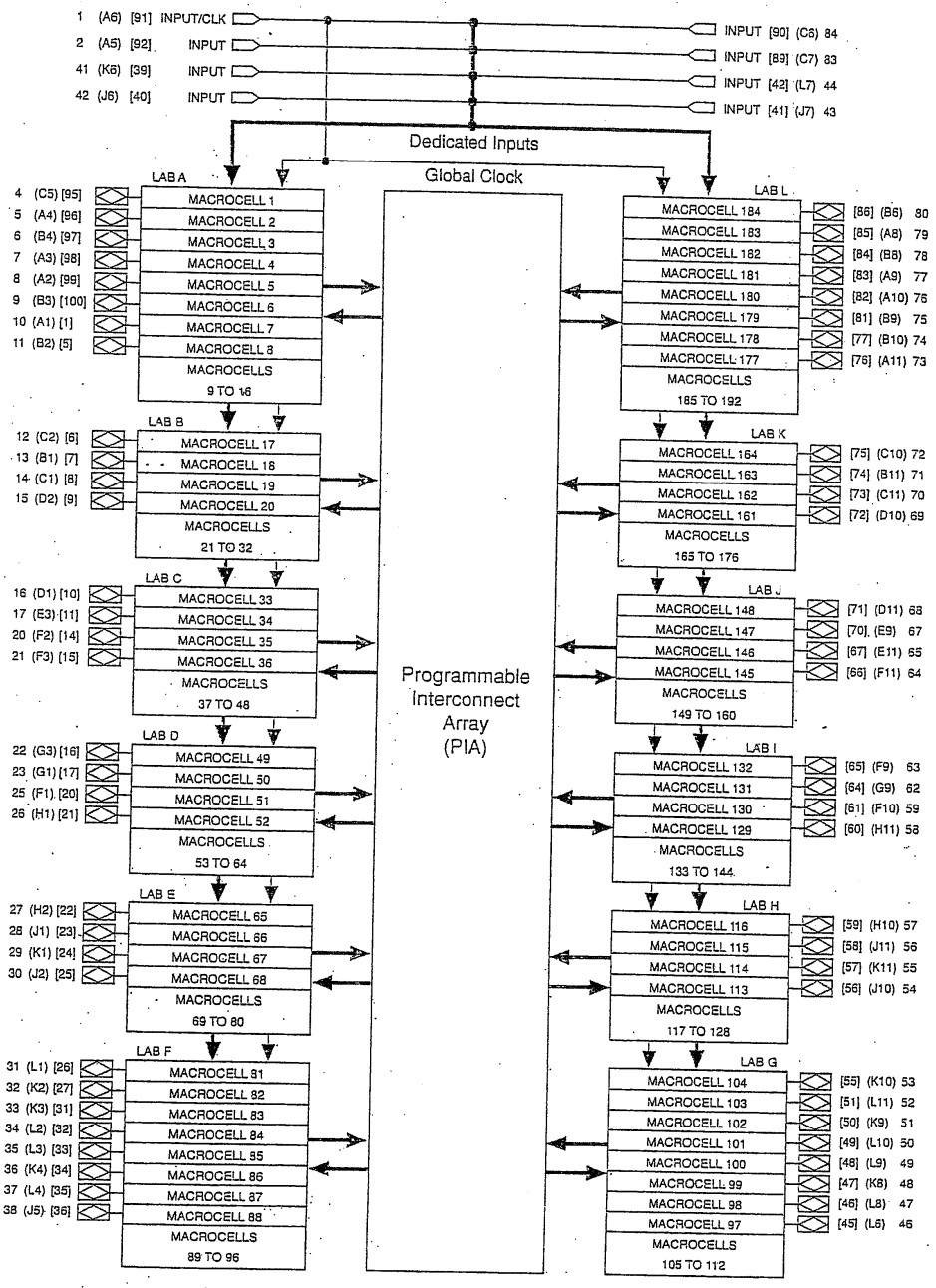
PROFESSORE: Roberto Saletti (http://unimap.unipi.it/cercapersone/dettaglio.php?ri=5663&template=dettaglio.tpl)

LINK AL CORSO ANNO 2017/2018: http://unimap.unipi.it/registri/dettregistriNEW.php?re=2087614::::&ri=8145

FREQUENTAZIONE: Consigliata.

Figure 26. EPM5192 Block Diagram

Numbers without parentheses are for J-lead packages. Numbers in parentheses are for PGA packages. Numbers in brackets are for EPM5192A QFP packages.



MACROCELLE 8+16 SONO DESTINATE A SVOLGERE FUNZIONI SEPOLTE

-> NO I/O

Figure 1. MAX 5000 Architecture

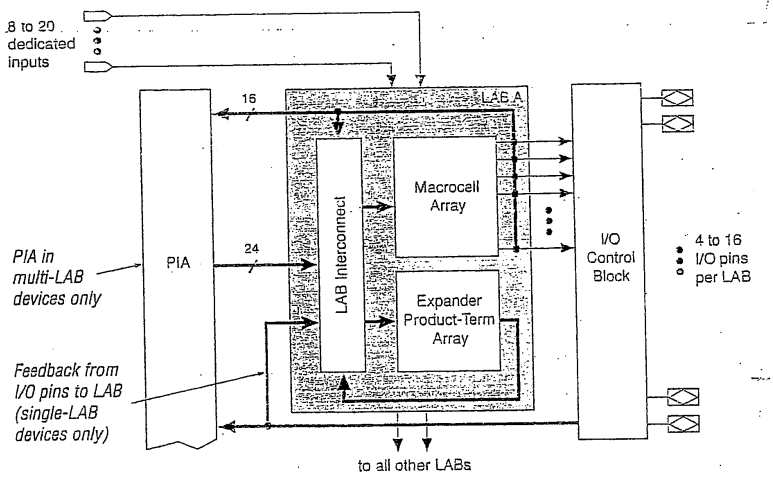


Figure 2. MAX 5000 Macrocell

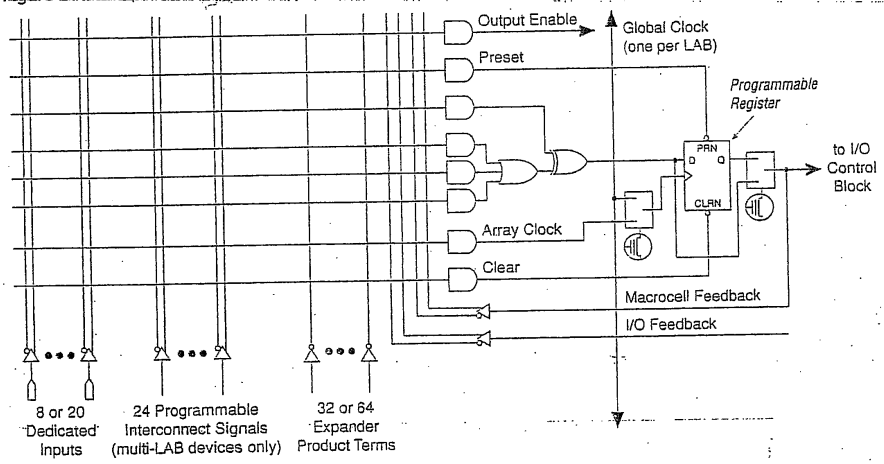
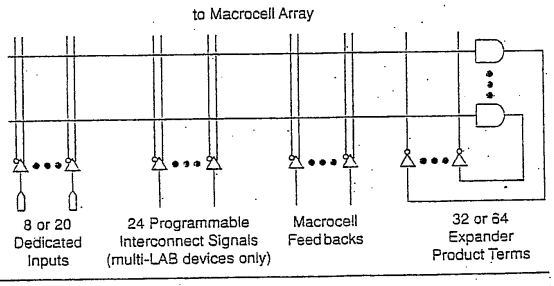


Figure 3. Expander Product Terms

Expander product terms are unallocated logic that can be used and shared by all macrocells in an LAB. Sharing allows efficient integration of complex combinatorial functions.



$y' = y \oplus \text{Polarità} = (\text{Somma di Prodotti}) \oplus (\text{Polarità})$

Questa consente di ottenere un livello di logica in più.

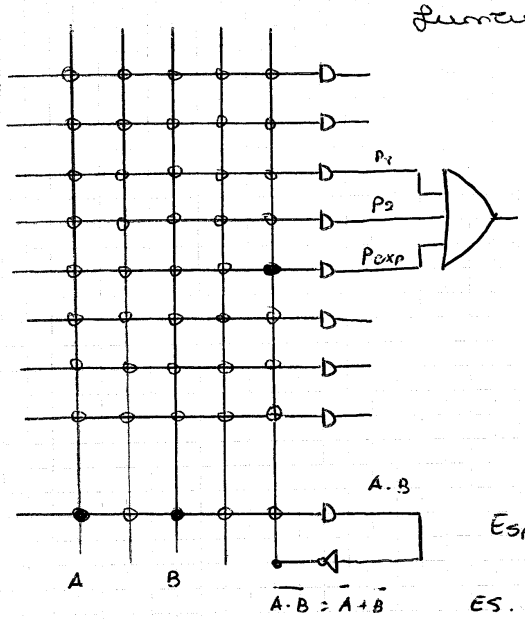
I MUX sono sempre pilotati da BIT di configurazione, mentre output-enable, Preset, Clear, Polarity sono definiti dal prodotto degli ingressi.

concludendo, l'architettura MAX5000 ha pochi termini di prodotto per costruire la funzione logica, diventa più importante avere flessibilità utilizzando termini di prodotto per servizio. Notevole vantaggio è introdotto, ulteriormente, dalla doppia reazione.

Se i termini di prodotto sono pochi, è necessario un numero di termini di prodotto > 3 è necessario usare ESPANSORI DI LOGICA

In fondo al piano ANS, ultimo Slice:

L'espansore di logica consente di inserire una somma di prodotti negata.
-> termine di prodotto libero condivisibile con altre funzioni.



Espansore di logica: termine di prodotto libero.

ES.

FAMIGLIA MAX 7000

Rispetto alla famiglia 5000 che ha 3 termini di prodotto e 5 termini di servizio si procede a costruire una nuova architettura.

Qui si hanno 5 termini di prodotto:

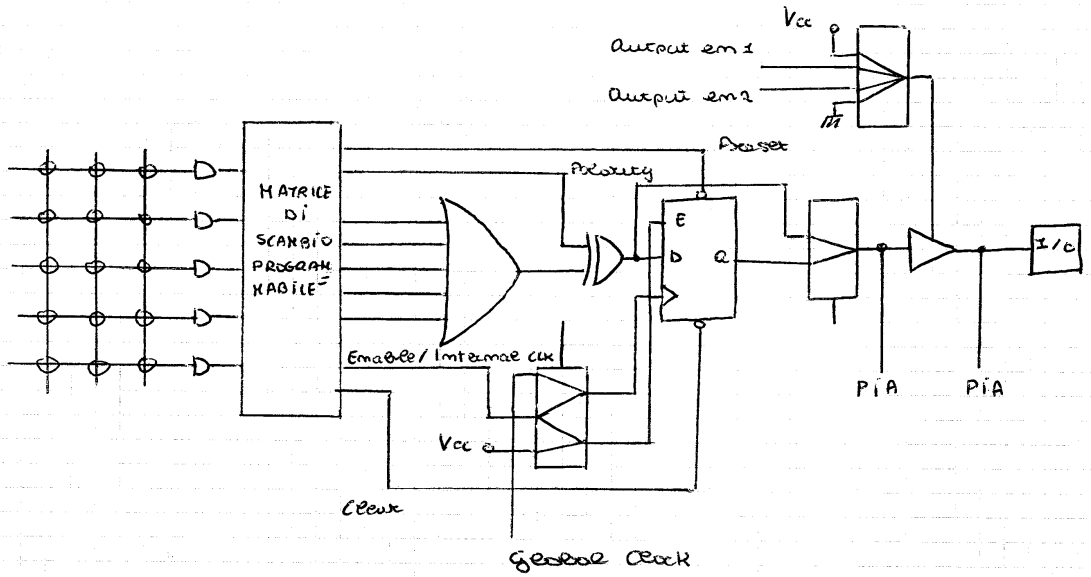


Figure 1. EPM7032, EPM7064 & EPM7096 Device Block Diagram

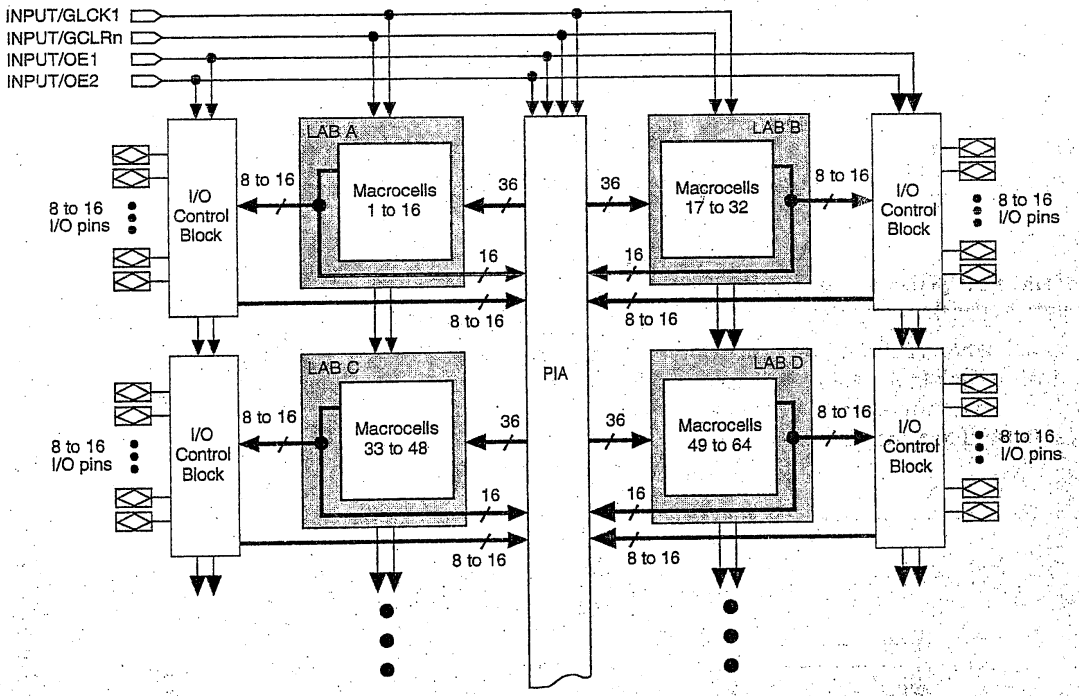


Figure 3. EPM7032, EPM7064 & EPM7096 Device Macrocell

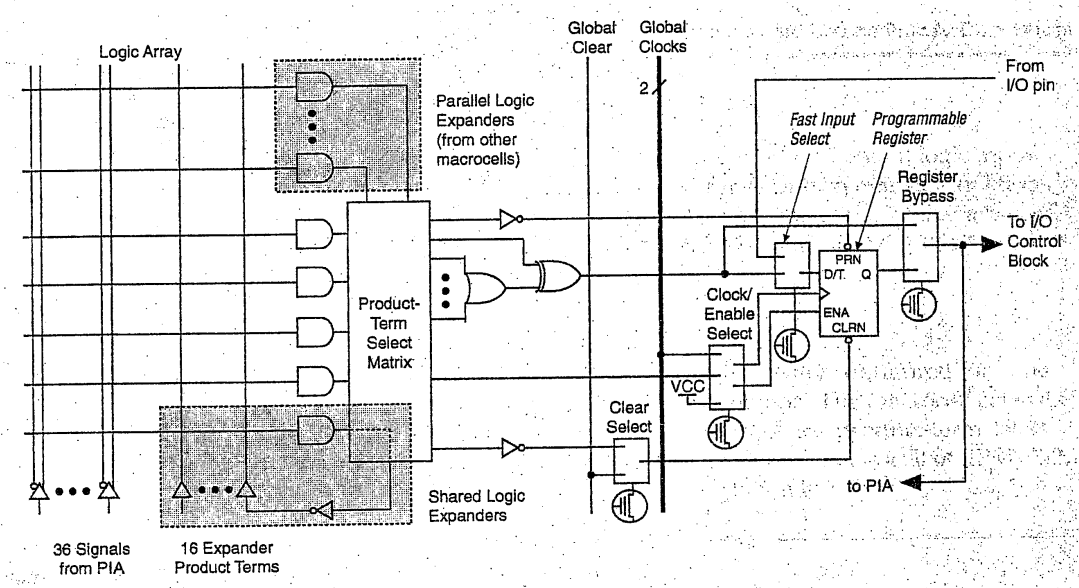


Figure 5. Shareable Expanders

Shareable expanders can be shared by any or all macrocells in an LAB.

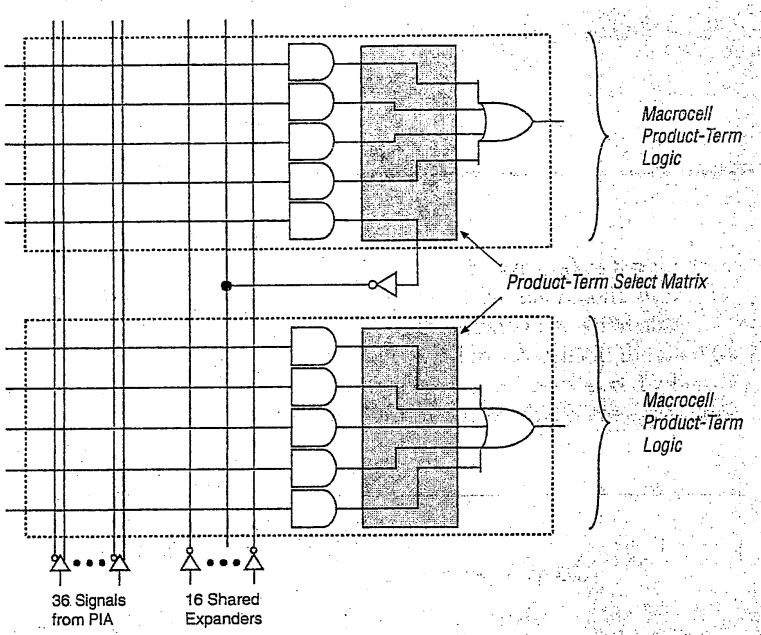
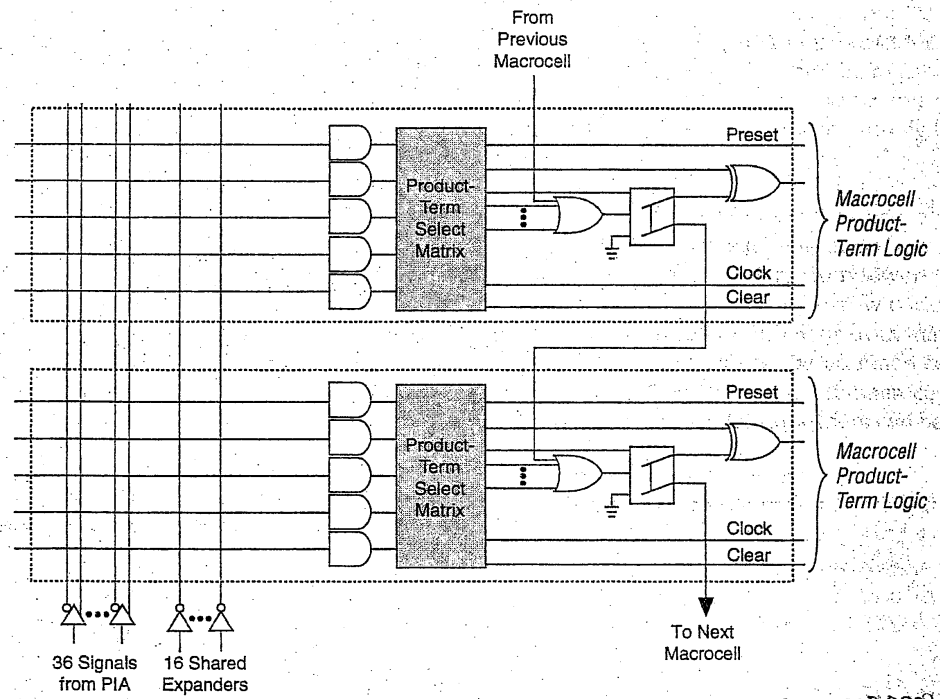


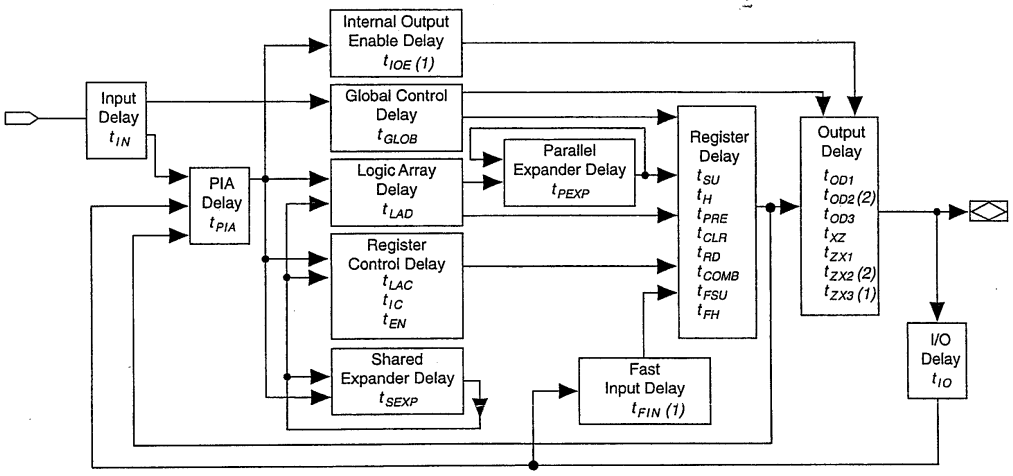
Figure 6. Parallel Expanders

Unused product terms in a macrocell can be allocated to a neighboring macrocell.



Questo metodo di espansione consente di ottenere un ritardo minore nella RC in quanto evita tutta la fatica che non mi interessa

Figure 12. MAX 7000 Timing Model



Notes:
(1) Only available in MAX 7000E and MAX 7000S devices.
(2) Not available in 44-pin devices.

The timing characteristics of any signal path can be derived from the timing model and parameters of a particular device. External timing parameters, which represent pin-to-pin timing delays, can be calculated as the sum of internal parameters. Figure 13 shows the internal timing relationship of internal and external delay parameters.

Figure 2. Multi-LAB MAX 5000 Device Timing Model

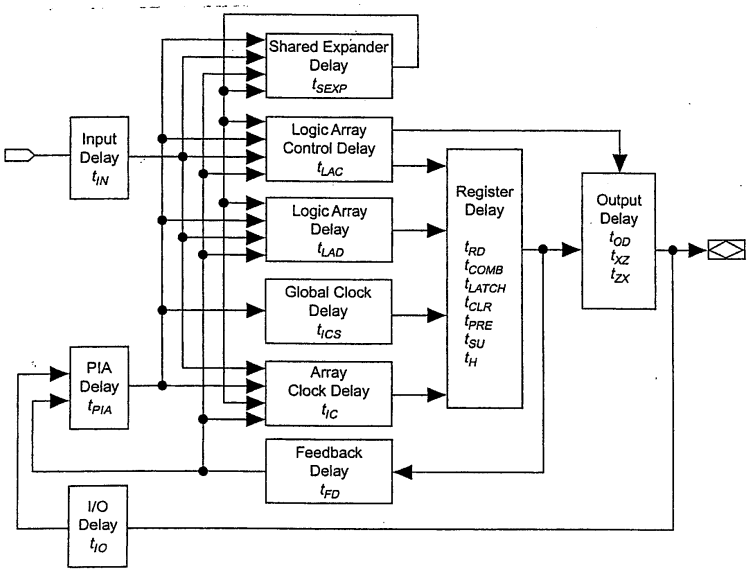
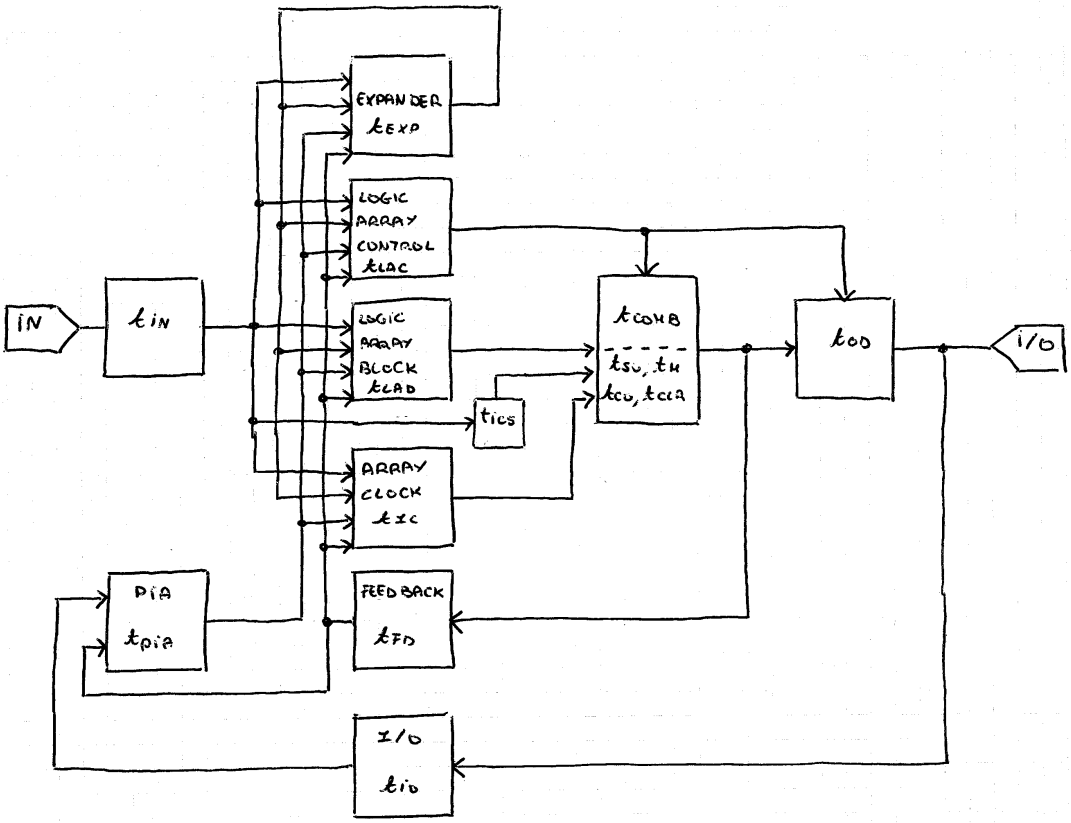


Table 17. MAX 7000 & MAX 7000E Internal Timing Parameters *Note (1)*

Symbol	Parameter	Conditions	Speed Grade -6		Speed Grade -7		Unit
			Min	Max	Min	Max	
t_{IN}	Input pad and buffer delay			0.4		0.5	ns
t_{IO}	I/O input pad and buffer delay			0.4		0.5	ns
t_{FIN}	Fast input delay	*(2)*		0.8		1.0	ns
t_{SEXP}	Shared expander delay			3.5		4.0	ns
t_{PEXP}	Parallel expander delay			0.8		0.8	ns
t_{LAD}	Logic array delay			2.0		3.0	ns
t_{LAC}	Logic control array delay			2.0		3.0	ns
t_{IOE}	Internal output enable delay	*(2)*				2.0	ns
t_{OD1}	Output buffer and pad delay Slow slew rate = off, V_{CCIO} = 5.0 V	C1 = 35 pF		2.0		2.0	ns
t_{OD2}	Output buffer and pad delay Slow slew rate = off, V_{CCIO} = 3.3 V	C1 = 35 pF *(7)*		2.5		2.5	ns
t_{OD3}	Output buffer and pad delay Slow slew rate = on, V_{CCIO} = 5.0 V or 3.3 V	C1 = 35 pF *(2)*		7.0		7.0	ns
t_{ZX1}	Output buffer enable delay Slow slew rate = off, V_{CCIO} = 5.0 V	C1 = 35 pF		4.0		4.0	ns
t_{ZX2}	Output buffer enable delay Slow slew rate = off, V_{CCIO} = 3.3 V	C1 = 35 pF *(7)*		4.5		4.5	ns
t_{ZX3}	Output buffer enable delay Slow slew rate = on V_{CCIO} = 5.0 V or 3.3 V	C1 = 35 pF *(2)*		9.0		9.0	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		4.0		4.0	ns
t_{SU}	Register setup time		3.0		3.0		ns
t_{H}	Register hold time		1.5		2.0		ns
t_{FSU}	Register setup time of fast input	*(2)*	2.5		3.0		ns
t_{FH}	Register hold time of fast input	*(2)*	0.5		0.5		ns
t_{RD}	Register delay			0.8		1.0	ns
t_{COMB}	Combinatorial delay			0.8		1.0	ns
t_{IC}	Array clock delay			2.5		3.0	ns
t_{EN}	Register enable time			2.0		3.0	ns
t_{GLOB}	Global control delay			0.8		1.0	ns
t_{PRE}	Register preset time			2.0		2.0	ns
t_{CLR}	Register clear time			2.0		2.0	ns
t_{PIA}	PIA delay			0.8		1.0	ns
t_{LPA}	Low-power adder	*(8)*		10.0		10.0	ns

08/11/2017

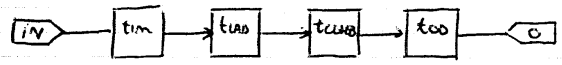
Descrizione temporale di una MAX 5000:



Le prestazioni di sistema emergono post fitting.
I microparametri qui presentati consentono di concludere un'analisi temporale post implementazione.

t_{IN}: ritardo ingresso - piano and
t_{LAD}: ritardo piano and - uscita porta OR
t_{COMB}: ritardo logica combinatoria
$t_{SU}, t_H, t_{CO}, t_{CLR}$: parametri logica sequenziale (FLIP FLOP)
t_{OD}: ritardo del blocco di uscita
t_{LAC}: ritardo piano and - segnali di controllo
t_{EXP}: ritardo piano and - piano and
t_{ICS}: ritardo global clock - FLIP FLOP
t_{IC}: ritardo clock interno - FLIP FLOP
t_{FD}: ritardo uscita - piano and
t_{PIA}: ritardo PIA
t_{IO}: ritardo pin I/O - ingresso PIA

Ritardo di Propagazione Combinatorio (T_{PD})

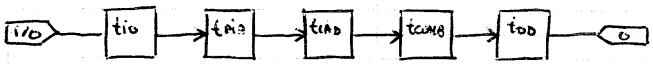


t: MICRO PARAMETRO

T: MACRO PARAMETRO

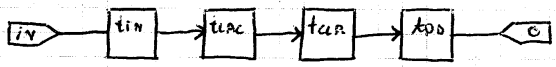
$$T_{PD} = t_{IN} + t_{LAD} + t_{COMB} + t_{OD}$$

Se il costruttore ha fornito i parametri definiti precedentemente, post fitting è possibile calcolare il t_{PD}, in quanto solo dopo l'implementazione su silicio è possibile conoscere il percorso tra IN ed O.



$$T_{PD} = t_{io} + t_{pia} + t_{LAD} + t_{comb} + t_{OD}$$

Ritardo Clear Flip Flop (T_{CLR})

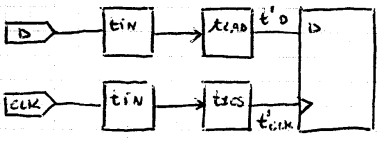


$$T_{CLR} = t_{IN} + t_{LAC} + t_{CLR} + t_{OD}$$

Se giungo dal piedino I/O:

$$T_{CLR} = t_{IO} + t_{PIA} + t_{LINE} + t_{CLR} + t_{OD}$$

Tempo di Set Up della struttura (T_{SU})



t_{CLK} : istante in cui c'è un fronte esterno di clk sul piedino clk

t_D : istante in cui varia D.

$t'_{CLK} = t_{CLK} + t_{IN} + t_{ICS}$

$t'_D = t_D + t_{IN} + t_{LAD}$

Da cui risulta: $t_D' \leq t_{CLK}' - t_{SU}$

$t_D + t_{IN} + t_{LAD} \leq t_{CLK} + t_{IN} + t_{ICS} - t_{SU}$

$t_D \leq t_{CLK} - t_{LAD} + t_{ICS} - t_{SU}$

Ovvero dall'esterno devo rispettare:

$t_D \leq t_{CLK} - T_{SU}$

con $T_{SU} = t_{LAD} - t_{ICS} + t_{SU}$

Tempo di Hold della struttura (T_H)

$t_D'' \geq t'_{CLK} + t_H$

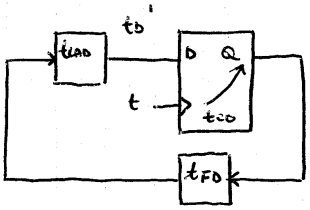
t_D'': istante in cui viene rimosso t_D'

$t_D'' = t_D + t_{IN} + t_{LAD} \geq t'_{CLK} + t_H = t_{CLK} + t_{IN} + t_{ICS} + t_H$

$t_D + t_{IN} + t_{LAD} \geq t_{CLK} + t_{IN} + t_{ICS} + t_H$

$t_D \geq t_{CLK} + t_{ICS} + t_H - t_{LAD} \rightarrow T_H = t_{ICS} + t_H - t_{LAD}$

Tempo di refresh dello stato del registro:



t: istante in cui c'è un fronte di clk sul registro

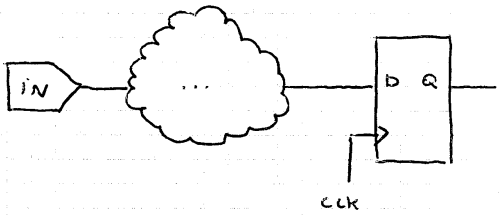
$t_D' = t + t_{CO} + t_{FD} + t_{LAD} \leq t + T_{CLK} - t_{SU}$ $\quad (t_D' \leq t + T_{CLK} - t_{SU})$

$T_{CLK} \geq t_{CO} + t_{FD} + t_{SU} + t_{LAD}$; $f_{CLK} = 1/T_{CLK}$

Si conclude che avendo a disposizione il modello dell'architettura e conoscendo il percorso seguito dal segnale siamo in grado di valutare i macro parametri T ricavando quindi i percorsi critici ottenendo, per esempio, $f_{CLK\,MAX}$.

10/11/2017

Per ciascun ingresso è quindi possibile associare un T_{SU} ed un T_H.



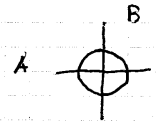
Il timing analyzer valuta tutti i percorsi e definisce il percorso critico.

Post fitting è possibile introdurre vincoli temporali in modo che lo strumento di sintesi sintetizzi l'architettura al fine di rispettarli.

Specifiche temporali: Timing Constraints.

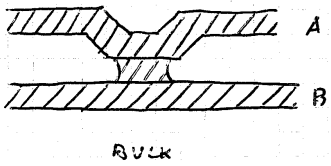
FPGA

Programmazione del collegamento:



Tecnologia Antifuse: dispositivo normalmente aperto che si chiude in seguito ad una programmazione.

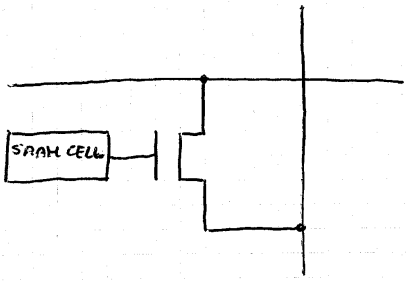
Armature separate da uno strato di ossido che collassano l'una sull'altra all'applicazione di un campo elettrico elevato.



a seguito dell'applicazione di un campo elettrico elevato

Questa è una programmazione permanente non reversibile. Di conseguenza è una tecnologia non volatile.

Tecnologia a MOS SRAM based.



a seconda del valore contenuto nella cella SRAM si decide se presente o meno del collegamento.

Ogni volta che il dispositivo perde l'alimentazione, l'informazione contenuta nella SRAM viene persa.

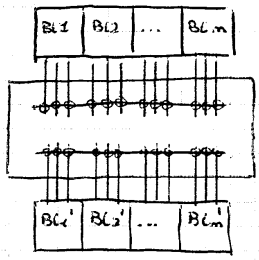
Tecnologia volatile

Anti Fuse	SRAM cell
Irreversibile, non volatile	Volatile, reversibile
Solida nei confronti delle interferenze e disturbi	100% testata e funzionante → affidabilità = 100%
affidabilità < 100%.	Sensibile ai disturbi
One Time Programmable (OTP)	Riprogrammabile
Richiede passi di processo appositi.	Riconfigurabile Run Time

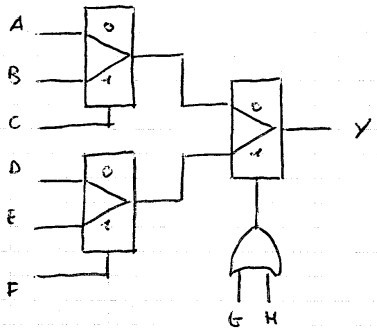
SRAM cell richiede di essere programmata all'accensione del dispositivo → necessita sempre di una fase di upload della configurazione

Architetture FPGA

Architettura Multiplexermi Actel ACT 1



Struttura di Bi (Blocco Logico)



Struttura base ad MUX

È una struttura non programmabile.

$Y = (G+H)(D\bar{F} + EF) + (\overline{G+H})(A\bar{C} + BC)$

Dove sta la programmabilità ?

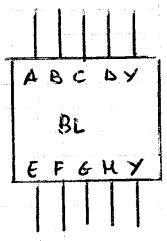
$Y = \overline{X_1 + X_2 + X_3} = \bar{X_1} \cdot \bar{X_2} \cdot \bar{X_3}$

$Y = (G+H)(D\bar{F} + EF) + \bar{G}\bar{H}A\bar{C} + \bar{G}\bar{H}BC = \bar{G}\bar{H}\bar{C}$

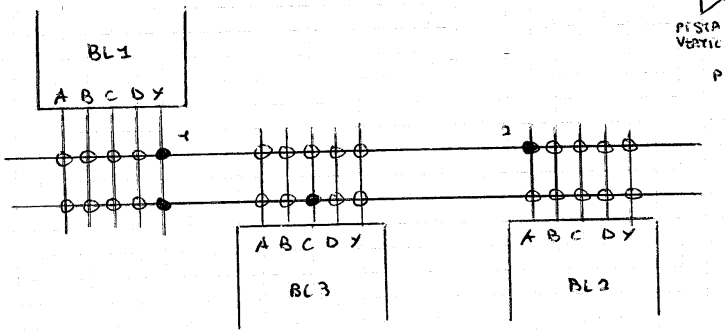
A = 1
B = Ø
C = X_3
D = Ø
E = Ø
F = X
G = X_1
H = X_2

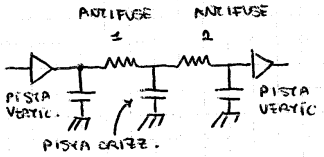
La programmazione si riduce in programmazione delle connettività e non nella programmazione della logica.

Descrizione del blocco:



Esempio



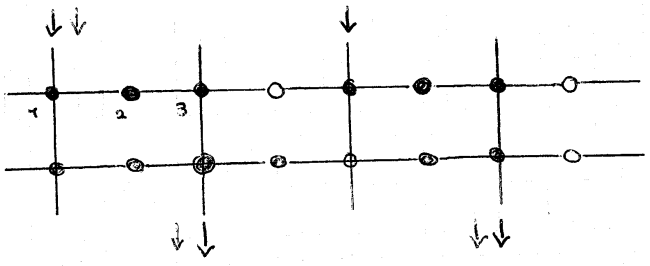


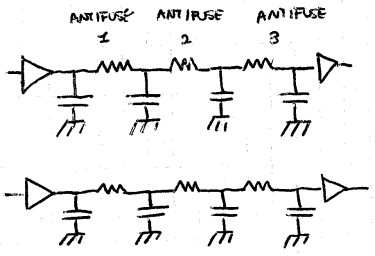
Ho collegato
$Y_1 = A_2$

Ho collegato
$Y_1 = C_3$

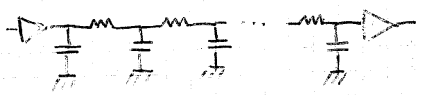
Il problema è che consuma immediatamente le risorse di comunicazione.

Prima soluzione: Segmentazione della linea.





Percorso generico



Rispetto all'architettura senza segmentazione abbiamo un incremento dei gruppi CA.

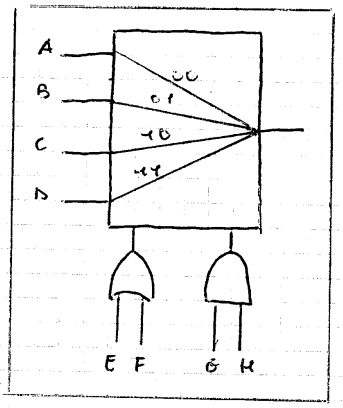
→ questo provoca un aumento del tempo di propagazione.

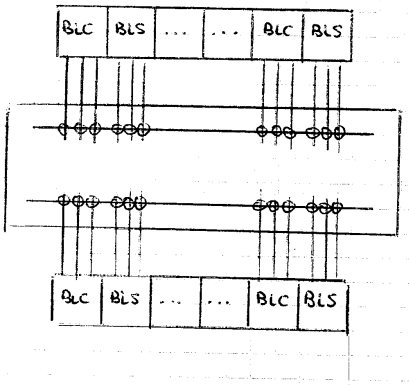
Soluzione: segmentazione parziale dove più utile nei cammini di routing.

Questo dispositivo non integra logica sequenziale: se necessaria un FLIP FLOP andrebbe costruita da una funzione logica.

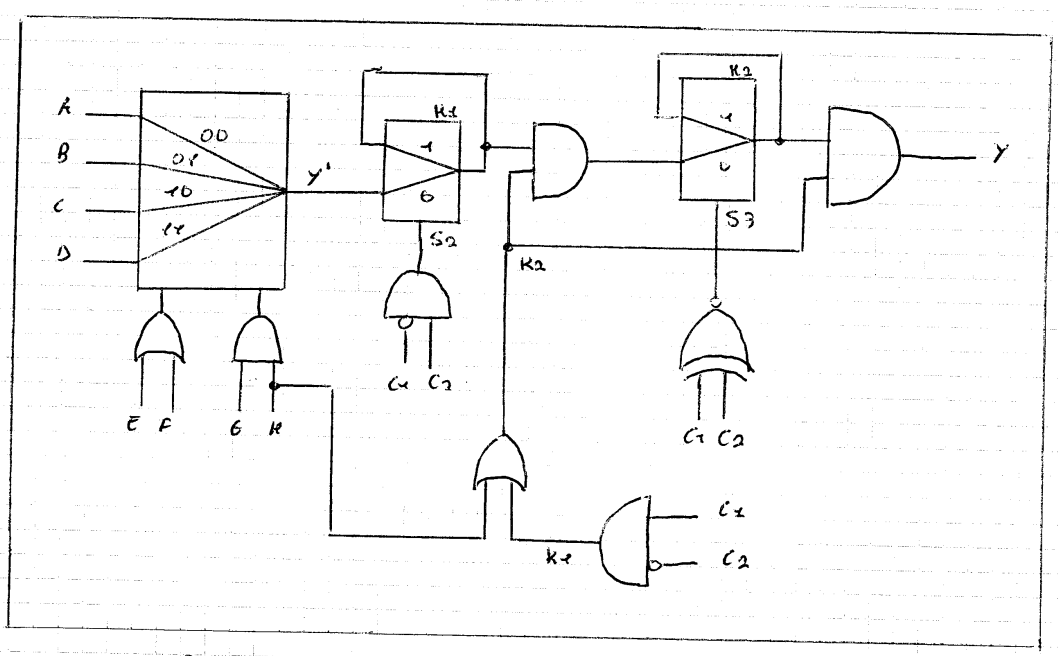
Aztec ACT 2

Struttura dei blocchi:





Blocco C



Blocco S

40MX and 42MX Automotive FPGA Families

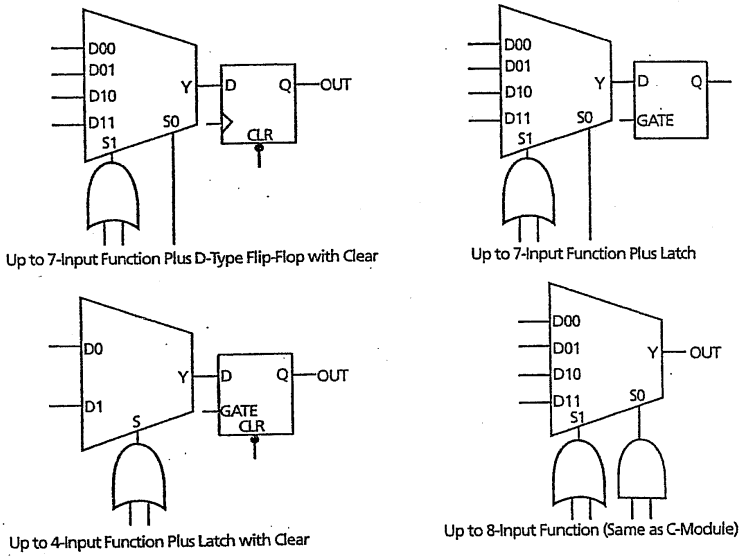


Figure 1-3 • **42MX S-Module Implementation**

MX Architectural Overview

The MX devices are composed of fine-grained building blocks that enable fast, efficient logic designs. All devices within these families are composed of logic modules, I/O modules, routing resources and clock networks, which are the building blocks for fast logic designs. In addition, the A42MX36 device contains embedded dual-port SRAM modules, which are optimized for high-speed datapath functions such as FIFOs, LIFOs and scratchpad memory. A42MX24 and A42MX36 also contain wide-decode modules.

Logic Modules

The 40MX logic module is an eight-input, one-output logic circuit designed to implement a wide range of logic functions with efficient use of interconnect routing resources (Figure 1-1).

The logic module can implement the four basic logic functions (NAND, AND, OR and NOR) in gates of two, three, or four inputs. The logic module can also implement a variety of D-latches; exclusivity functions, AND-ORs and OR-ANDs. No dedicated hardwired latches or flip-flops are required in the array; latches and flip-

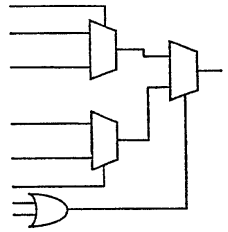


Figure 1-1 • **40MX Logic Module**

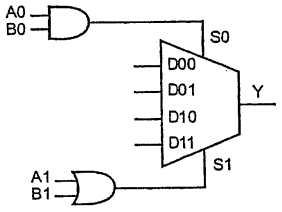


Figure 1-2 • **42MX C-Module Implementation**

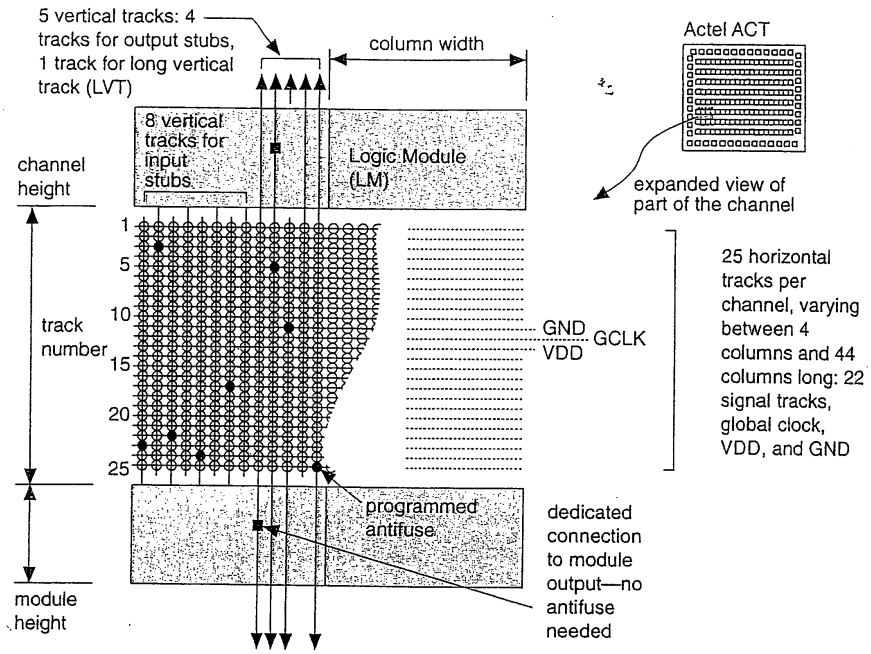


FIGURE 7.2 ACT 1 horizontal and vertical channel architecture. (*Source:* Actel.)

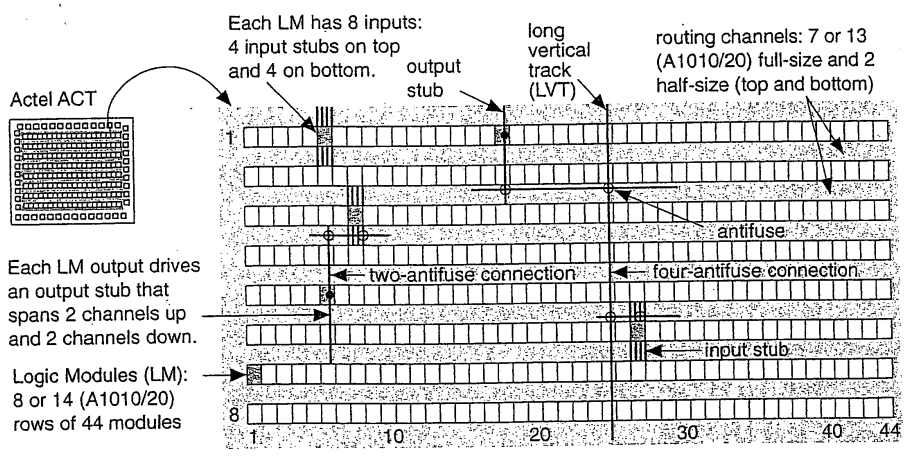


FIGURE 7.1 The interconnect architecture used in an Actel ACT family FPGA. (*Source:* Actel.)

$C_1 = 1$, $C_2 = \emptyset$:

-> $K_1 = 1$, $K_2 = 1$

$S_2 = \emptyset$, $S_3 = \emptyset$

$y = y'$ Blocco S configurato come un Blocco C.

$C_1 = \emptyset$, $C_2 = T$, T segnale esterno

-> $K_1 = \emptyset$, $K_2 = H$

$S_2 = C_2 = T$, $S_3 = \bar{C}_2 = \bar{T}$

Suppongo $T = \emptyset$: $Y = y'$

Adesso $T = \emptyset$ -> 1

H_1 si comporta come un LATCH con abilitazione attivo su $T = \emptyset$.

-> mantiene y'

Suppongo $H = 1$;

H_2 mantiene : si comporta come un LATCH con abilitazione attivo su $T = \emptyset$ ($\bar{T} = 1$)

È un'architettura master_slave : Flip Flop D Positive Edge.

T è il CLOCK. Blocco S : logica + FLIP FLOP D

$H = \emptyset$ è il reset dell'uscita.

$C_1 = T$, $C_2 = \bar{T}$

-> $K_1 = T$, $K_2 = T + H$ ($H = 1$)

$S_2 = \bar{T}$ ($S_2 = \bar{C}_1 . C_2 = \bar{T} . \bar{T} = \bar{T}$)

$S_3 = \emptyset$

H_1 si comporta come un LATCH D attivo su $T = \emptyset$ ($\bar{T} = 1$).

Tutta l'architettura è un LATCH di tipo D attivo su $T = \emptyset$.

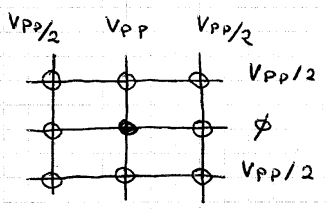
Blocco S : logica + LATCH D

15/11/2017

Antifusibile:

per programmarlo è necessario applicare una tensione elevata.

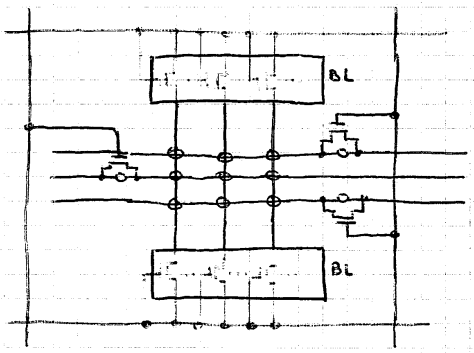
Ma come si programma un determinato antifusibile?



Con questo modo riesco a fondere l'antifuse.

V_{PP}: tensione di fusione.

Bisogna però ricordare che queste matrici non esistono nell'interno del circuito.



Questa è la struttura reale.

Per accedere agli antifuse aggiungo un passo transistor in parallelo alla segmentazione.

Aggiungo inoltre una colonna di programmazione per BL.

FPGA Xilinx Serie 3000

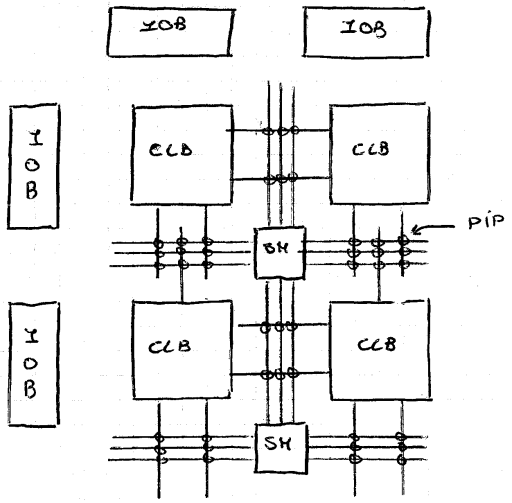
Programmazione basata su celle SRAM.

Precedentemente abbiamo visto logica programmabile con un piano AND o con un'architettura basata su MUX. Spesso si usa una LUT (LOOK UP TABLE): è sempre una memoria programmabile con la tabella di verità della funzione logica. -> gli indirizzi della LUT sono gli ingressi del sistema.

Consente di ottenere una funzione logica combinatoria qualunque con un numero di ingressi definito.

La LUT è una piccola PROM.

Architettura :

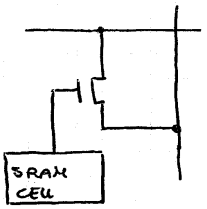


CLB : Combinatory Logic Block

IOB : I/O Block

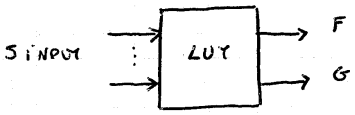
SH : Switch Matrix

PIP : Programmable Interconnect Point

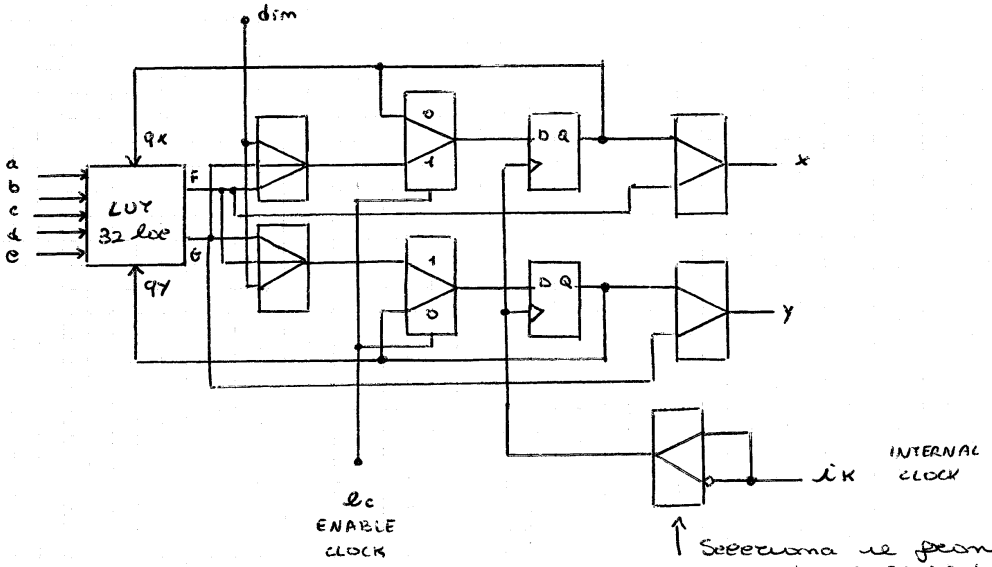


Organizzata in matrice non più per righe.

Struttura CLB :



Si può realizzare una funzione logica a 32 bit o 2 funzioni logiche a 16 bit.



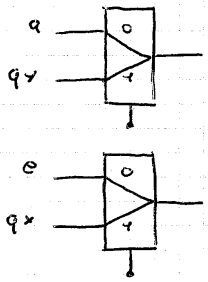
Seleziona il fronte di salita selezionato dal flip flop. (POS o NEG)

qx : restituisce x

qy : restituisce y

Posso generare due funzioni x ed y completamente indipendenti.

La LUT permette a 32 bit, quindi 5 ingressi: se includo qx e qy devo escludere 2 ingressi tra a, b, c, d, e.

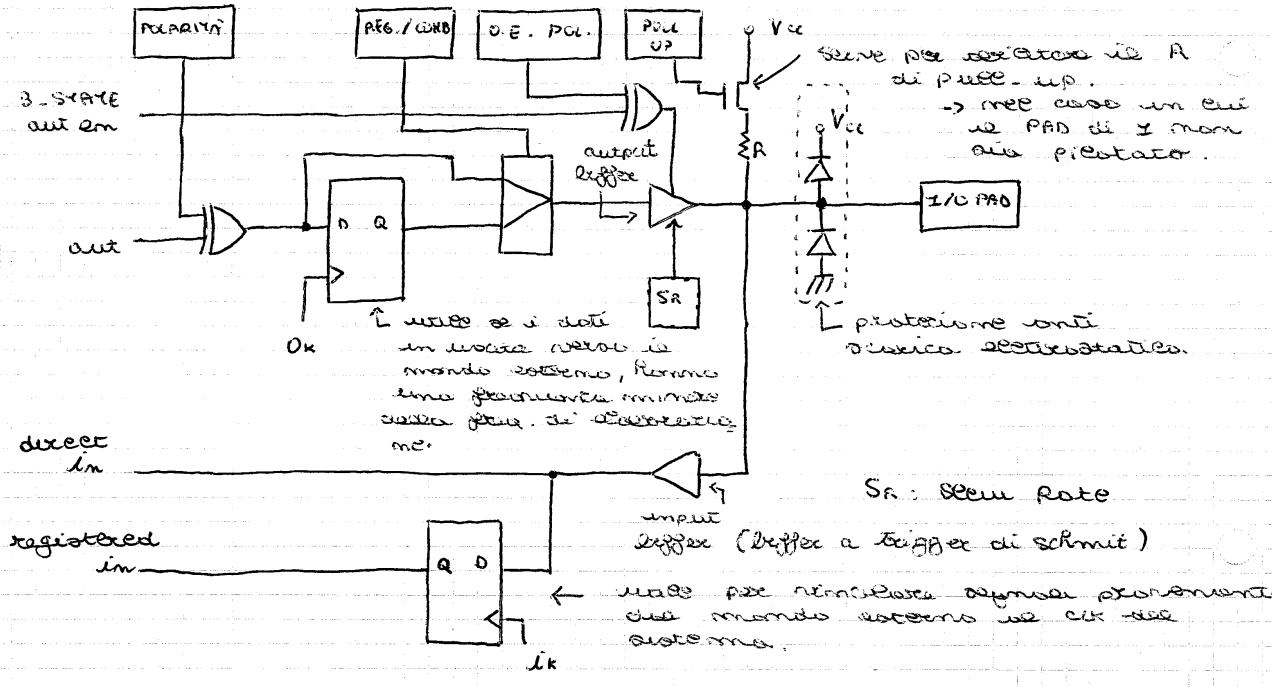


DIN consente di memorizzare un dato proveniente dall'esterno nella logica sequenziale del CLB.

Entra da din e non da a, b, c, d, e in modo da non incontrare il tempo di ritardo introdotto dalla LUT.

17/11/2017

Analisi IOB (Input Output Block)



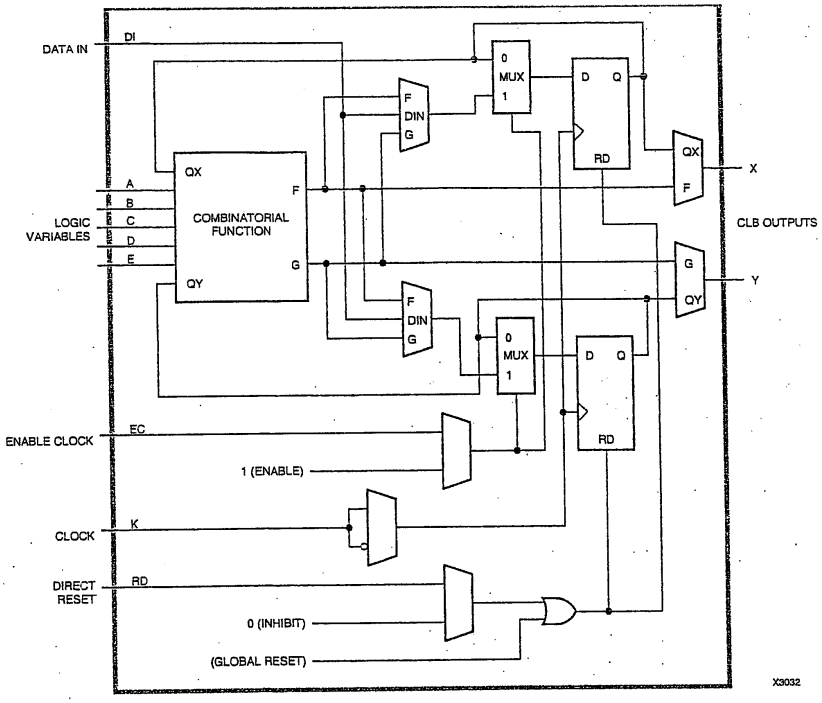


Figure 5: Configurable Logic Block.
Each CLB includes a combinatorial logic section, two flip-flops and a program memory controlled multiplexer selection of function. It has the following:

- five logic variable inputs A, B, C, D, and E
- a direct data in DI
- an enable clock EC
- a clock (invertible) K
- an asynchronous direct RESET RD
- two outputs X and Y

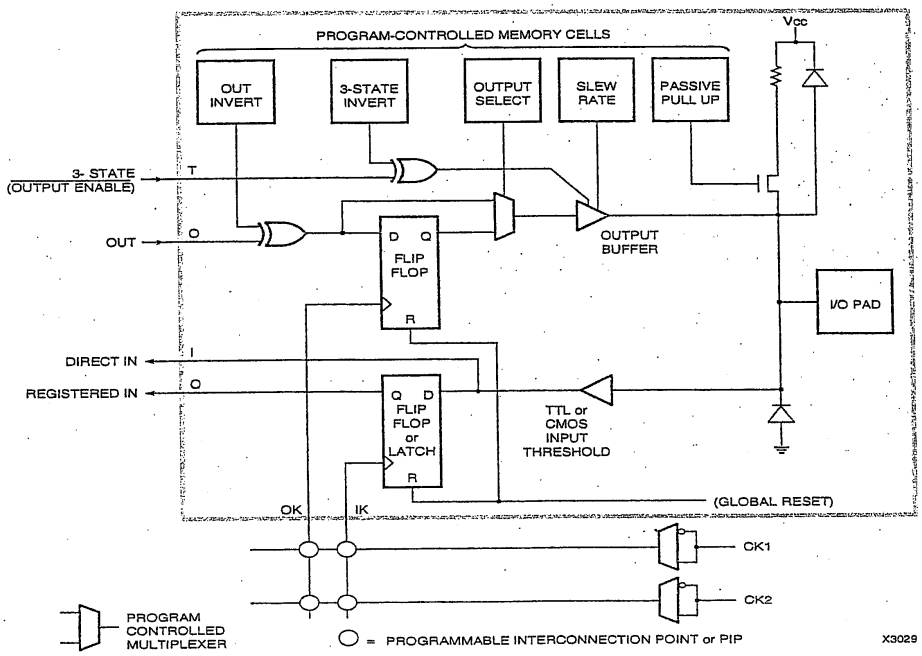


Figure 4: Input/Output Block.
Each IOB includes input and output storage elements and I/O options selected by configuration memory cells. A choice of two clocks is available on each die edge. The polarity of each clock line (not each flip-flop or latch) is programmable. A clock line that triggers the flip-flop on the rising edge is an active Low Latch Enable (Latch transparent) signal and vice versa. Passive pull-up can only be enabled on inputs, not on outputs. All user inputs are programmed for TTL or CMOS thresholds.

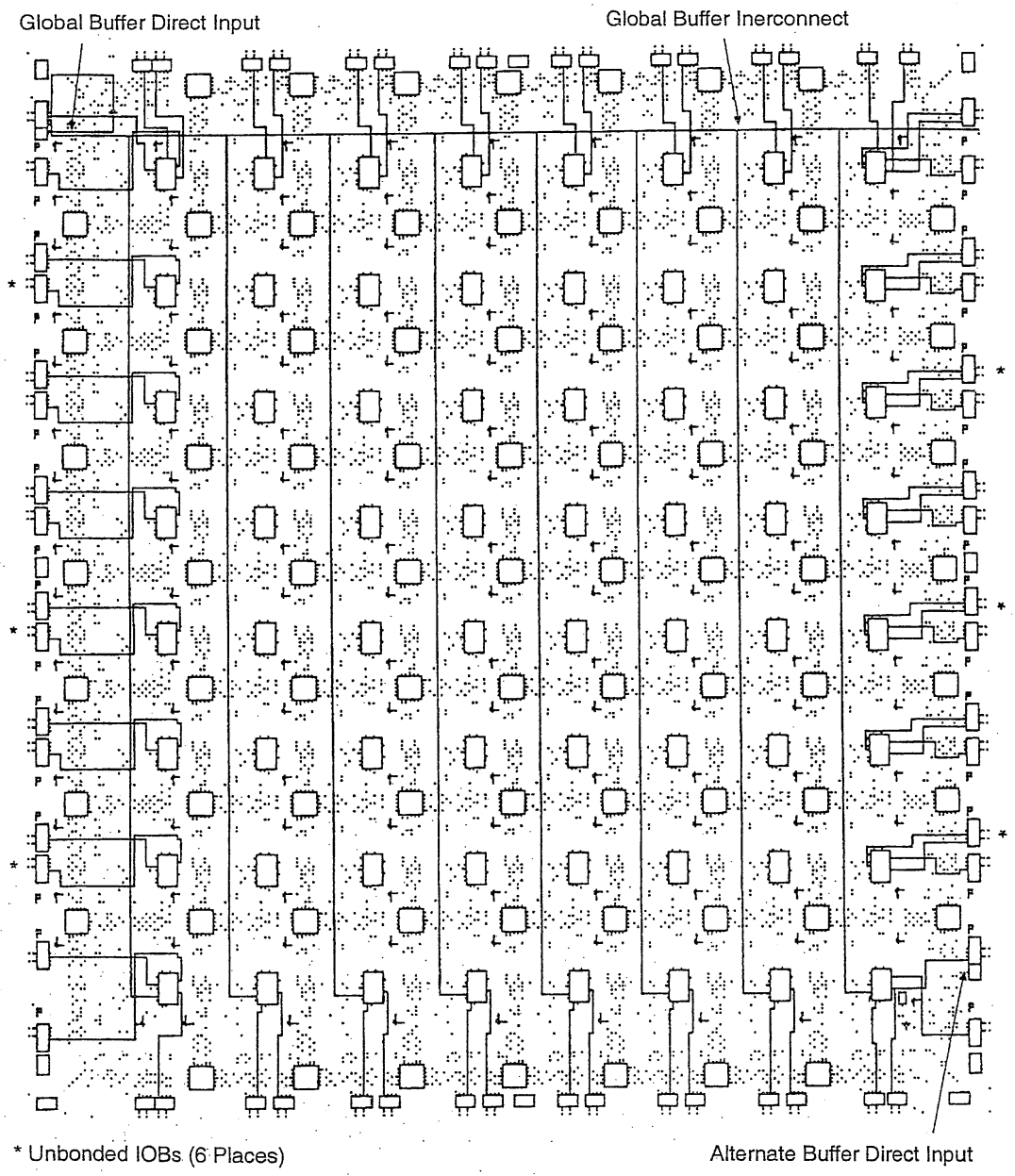


Figure 13: XC3020A Die-Edge IOBs. The XC3020A die-edge IOBs are provided with direct access to adjacent CLBs.

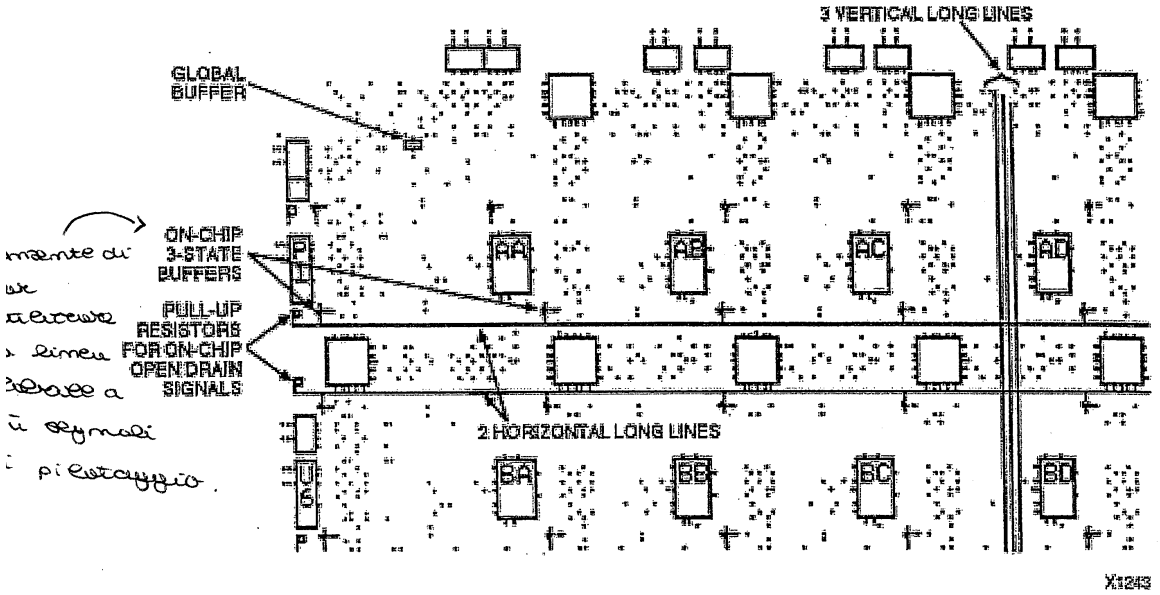


Figure 14: Horizontal and Vertical Longlines. These Longlines provide high fan-out, low-skew signal distribution in each row and column. The global buffer in the upper left die corner drives a common line throughout the FPGA.

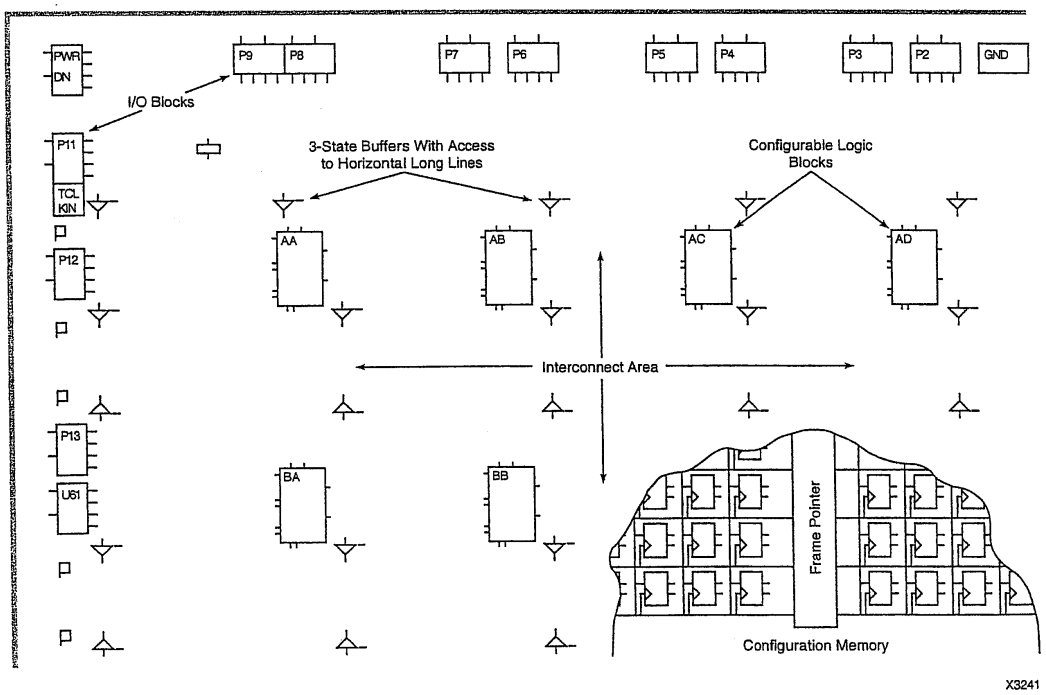


Figure 2: Field Programmable Gate Array Structure.
It consists of a perimeter of programmable I/O blocks, a core of configurable logic blocks and their interconnect resources. These are all controlled by the distributed array of configuration program memory cells.

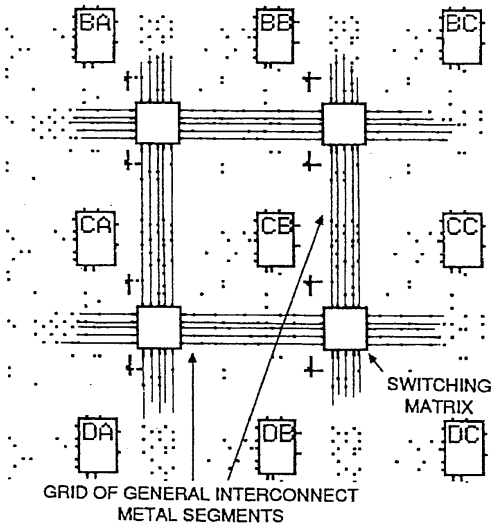


Figure 10: FPGA General-Purpose Interconnect.
Composed of a grid of metal segments that may be interconnected through switch matrices to form networks for CLB and IOB inputs and outputs.

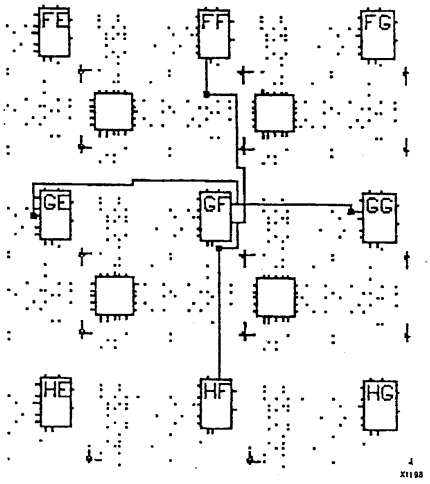
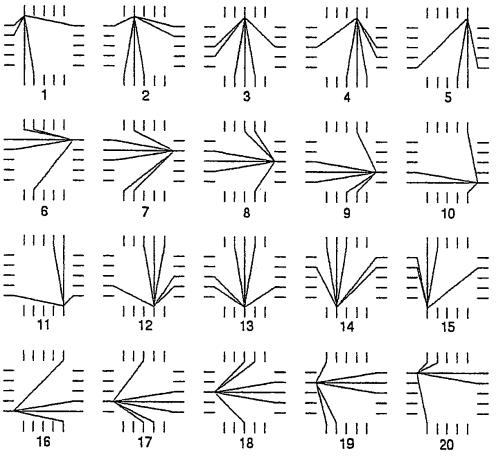


Figure 12: CLB X and Y Outputs.
The X and Y outputs of each CLB have single contact, direct access to inputs of adjacent CLBs

↑ ROUTING LOCALE

↑ ROUTING GENERALE



383 16

Figure 11: Switch Matrix Interconnection Options for Each Pin.
Switch matrices on the edges are different.

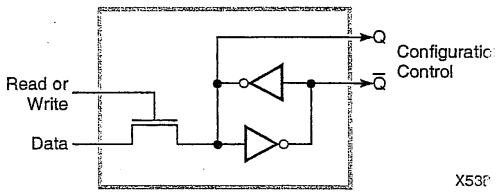
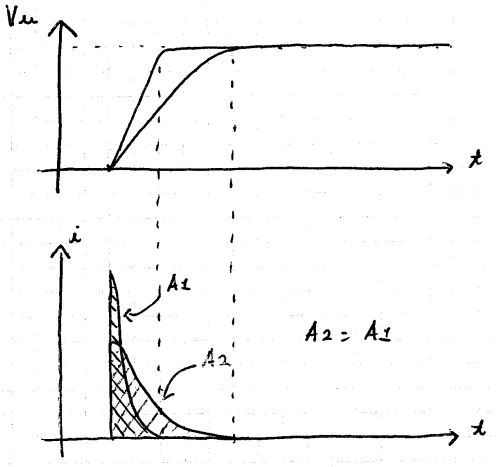


Figure 3: Static Configuration Memory Cell.
It is loaded with one bit of configuration program and c
trols one program selection in the Field Programmai
Gate Array.

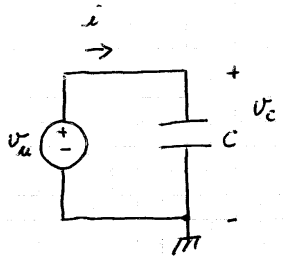
↑ Si nota che non tutti i PIN sono interconnessi.

Il bit SR mi consente di variare la slew rate del buffer: a parità di capacità di carico consente di ridurre il SETTLING TIME.
SR non fa altro che variare la corrente di polarizzazione del buffer: variano il numero di MOS ON. A parità di SETTLING TIME riesco a pilotare carichi di capacità elevata.

$P = C \cdot V_{cc}^2 \cdot f_{CLK}$: la SLEW RATE non influisce sulle performance energetiche del sistema
SR maggiore mi consente però di ridurre il SETTLING TIME avendo poter usare f_{CLK} maggiore.



- SR_1
- SR_2 con $SR_2 > SR_1$



$$i = C \cdot \frac{dV_C}{dt} = C \cdot \frac{dV_u}{dt}$$

Se aumento i picchi di corrente aumentando i disturbi sulla linea generando elettromagnetismo.

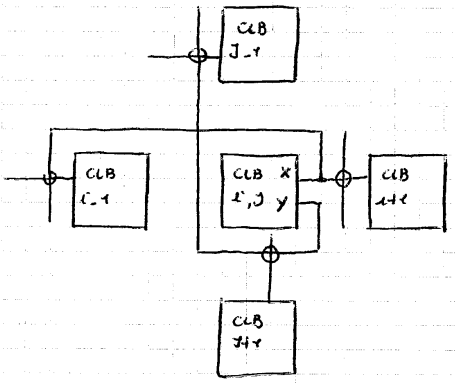
Generalmente in un FPGA è possibile connettere due parti qualunque: Routing generale attraverso canali di routing e programmazione connessioni e matrici di accesso.

I punti di connessione programmabili sono dei MOSFET attivi che introducono un effetto resistivo che assieme alla capacità delle linee introducono un ritardo associato al collegamento.

-> RITARDO LOGICA.

-> RITARDO COLLEGAMENTO.

In alcune topologie di FPGA sono presenti interconnessioni per routing locali.
Nella Xilinx XC3000 sono presenti e connessi alla porta X del CLB per le righe e alla porta Y del CLB per le colonne.



1 sola connessione programmabile per raggiungere le risorse immediatamente adiacenti.

Il routing generale è riservato per connessioni che attraversano grandi porzioni del circuito (es. CLK, RESET).
Per segnali specifici e non generali di sistema Risorse Dedicate (es. CLK)

Figure 2–1. Cyclone II EP2C20 Device Block Diagram

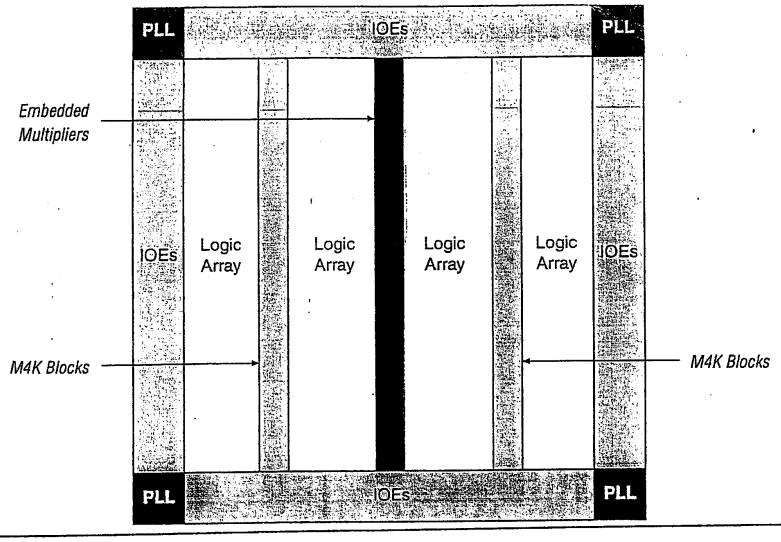


Figure 2–2. Cyclone II LE

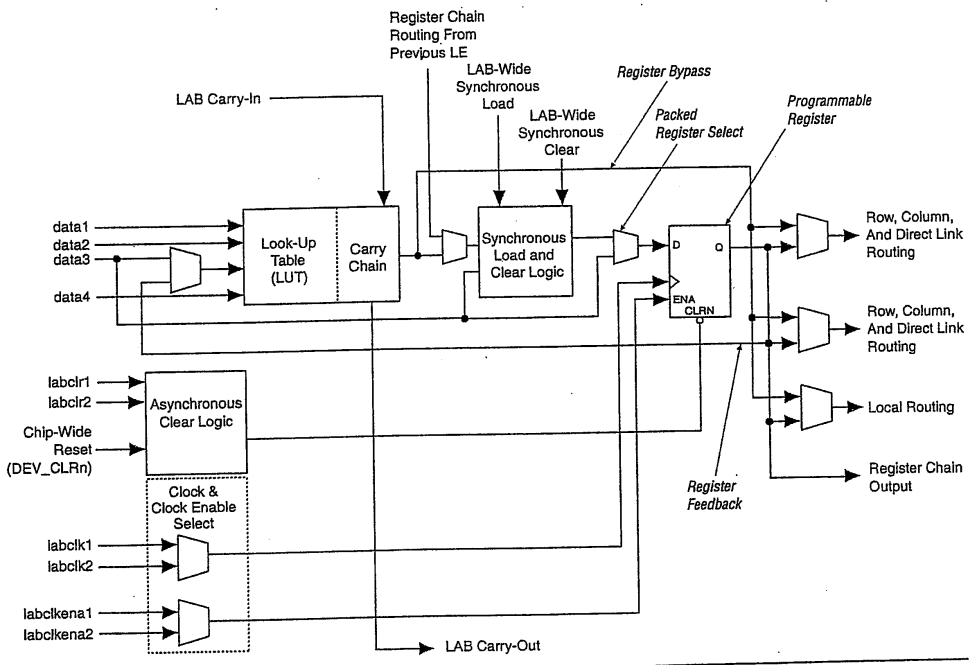


Figure 2–8. R4 Interconnect Connections

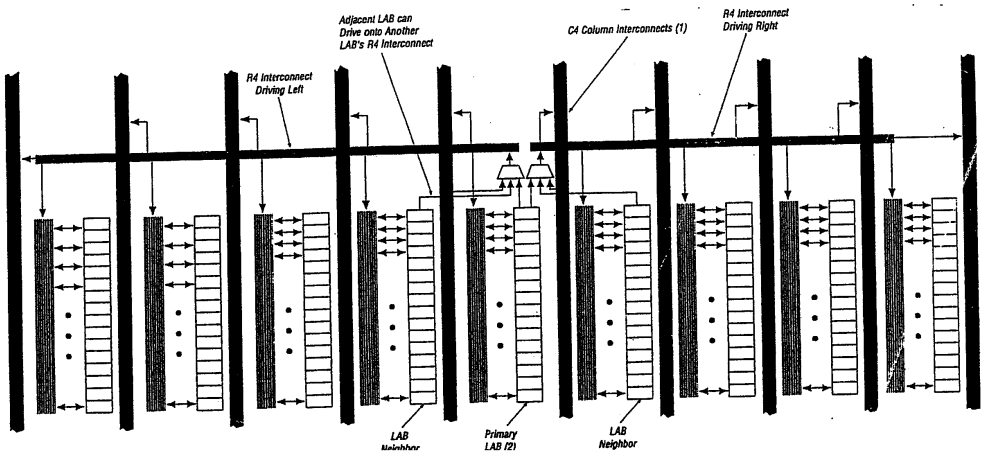


Figure 2–17. M4K RAM Block LAB Row Interface MEMORY BLOCK

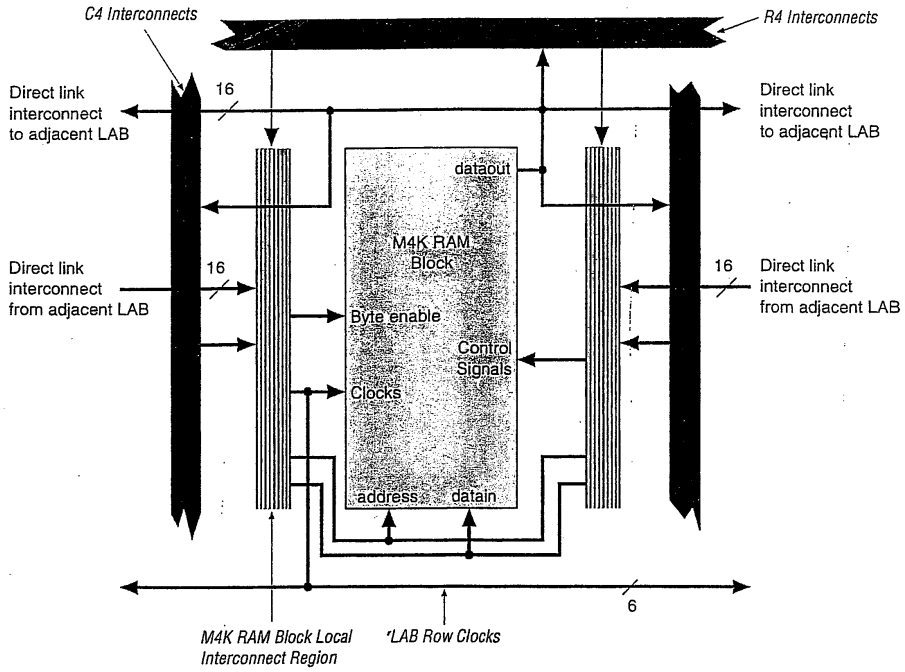


Figure 2–18. Multiplier Block Architecture

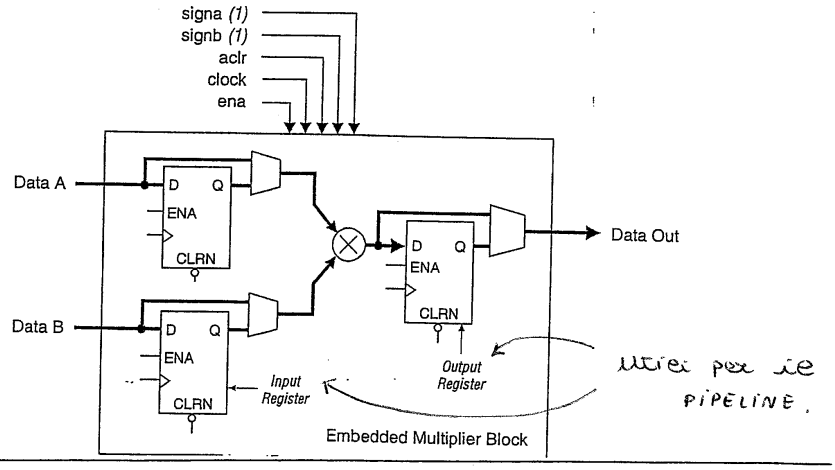
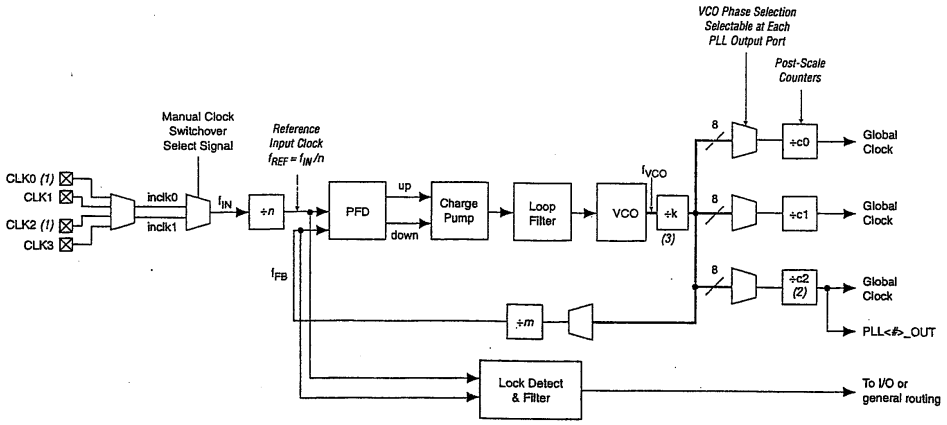


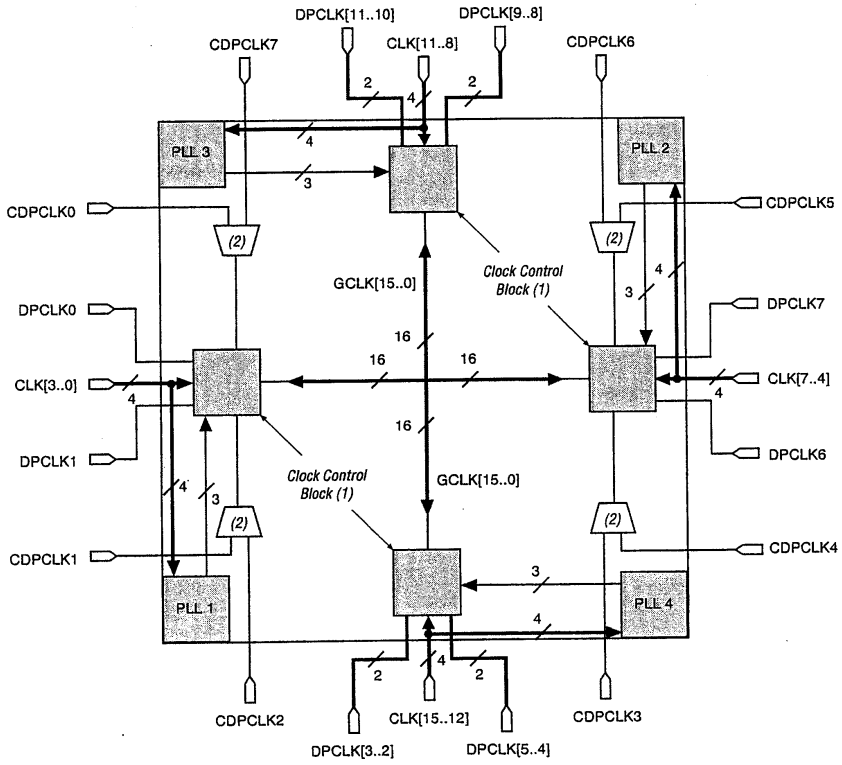
Figure 2–16. Cyclone II PLL *Note (1)*



Notes to Figure 2–16:

(1) This input can be single-ended or differential. If you are using a differential I/O standard, then two CLK pins are

Figure 2–12. EP2C15 & Larger PLL, CLK[], DPCLK[] & Clock Control Block Locations

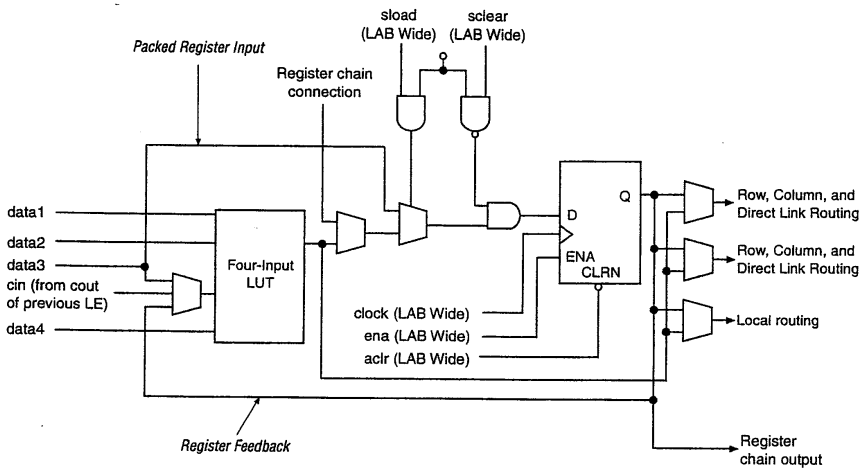


Notes to Figure 2–12:

(1) There are four clock control blocks on each side.

(2) Only one of the corner CDPCLK pins in each corner can feed the clock control block at a time. The other CDPCLK pins can be used as general-purpose I/O pins.

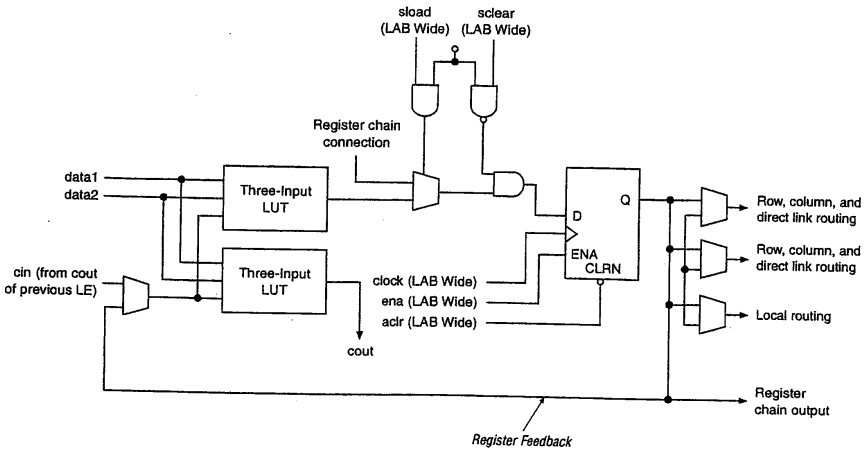
Figure 2–3. LE in Normal Mode



Arithmetic Mode

The arithmetic mode is ideal for implementing adders, counters, accumulators, and comparators. An LE in arithmetic mode implements a 2-bit full adder and basic carry chain (see Figure 2–4). LEs in arithmetic mode can drive out registered and unregistered versions of the LUT output. Register feedback and register packing are supported when LEs are used in arithmetic mode.

Figure 2–4. LE in Arithmetic Mode

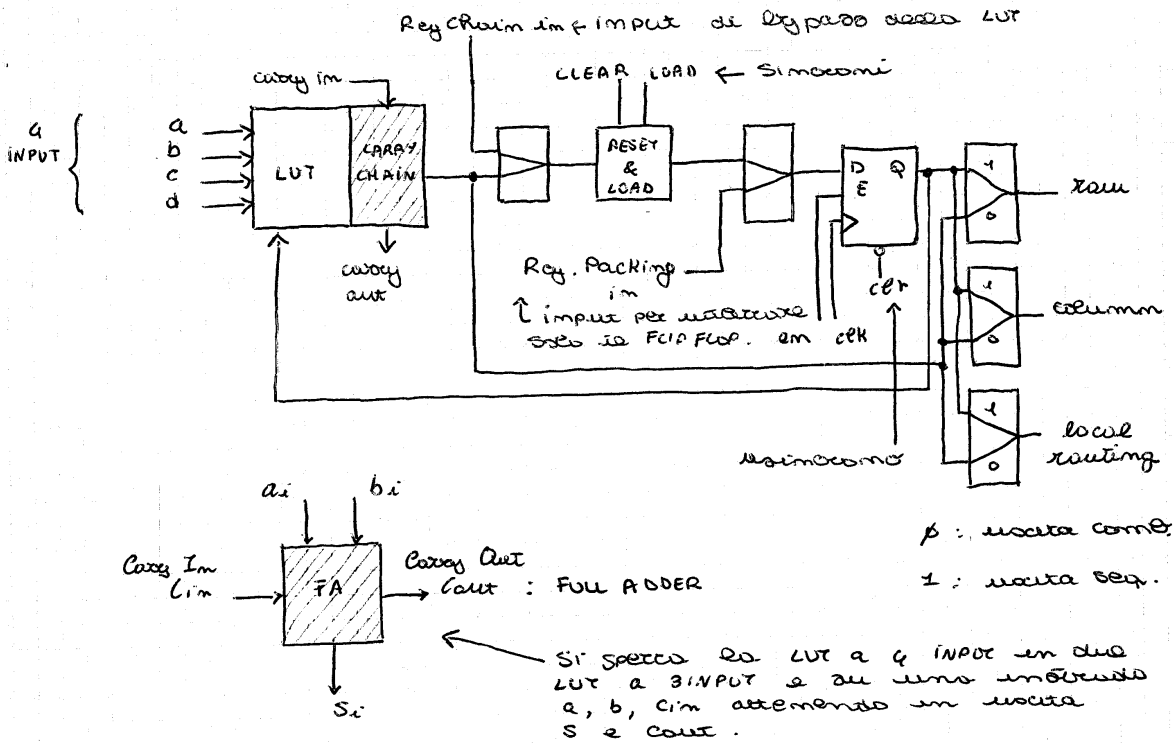


ALTERA CYCLONE II FPGA

Logica a look up tables.
Sono presenti però blocchi specifici che realizzano funzioni speciali come per esempio le EMBEDDED MEMORY BLOCK (risorse dedicate alla memorizzazione programmabile).
Sono presenti anche:
EMBEDDED MULTIPLIER (moltiplicatori programmabili)

Architettura:
Struttura basata su Logic Element (LE)

Struttura LE CYCLONE II:



Le CYCLONE II ha a bordo 4 PLL per generare clock diversi da quello generato dall'oscillatore quarzato.

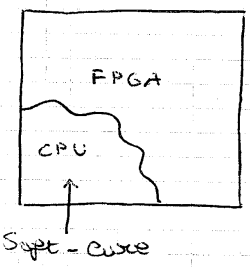
Oltre CYCLONE II:

La tendenza è avere all'interno degli FPGA strutture dedicate a svolgere particolari funzioni.

Si cerca di far collaborare la programmabilità HW con la programmabilità SW.

Si introduce all'interno dell FPGA un core processore ottenuto a design HDL sulle Risorse programmabile.

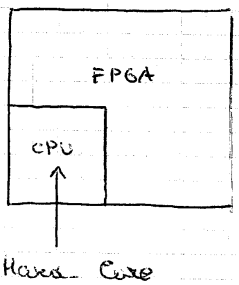
Una parte di HARDWARE verrà in ingresso dei segnali che vengono denominati istruzioni.



Abbiamo costruito un microcontrollore su misura per le nostre specifiche.

Soft - core è un core CPU ottenuto su FPGA.

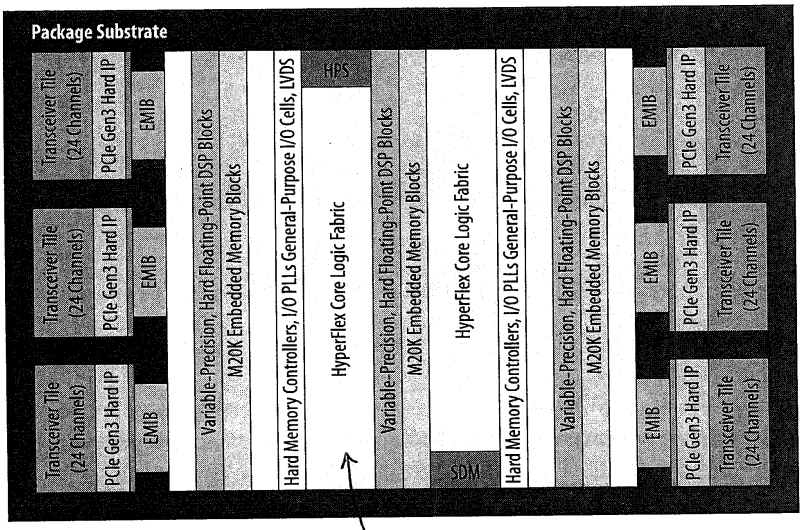
Se passo successivo è inserire un Hard - core ASIC accanto sul FPGA.



Hard - core è un core CPU ASIC, non ottenuto.

Stratix 10 Block Diagram

Figure 1: Stratix 10 FPGA and SoC Architecture Block Diagram



Adaptive Logic Module (ALM)

Stratix 10 devices use a similar adaptive logic module (ALM) as the previous generation Arria 10 and Stratix V FPGAs, allowing for efficient implementation of logic functions and easy conversion of IP between the devices.

The ALM block diagram shown in the following figure has eight inputs with a fracturable look-up table (LUT), two dedicated embedded adders, and four dedicated registers.

Figure 8: Stratix 10 FPGA and SoC ALM Block Diagram

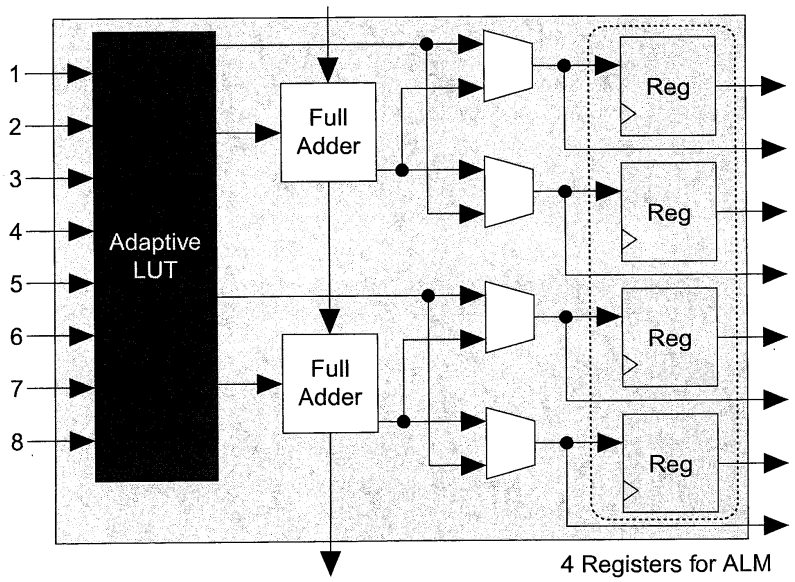


Figure 3: HyperFlex Core Architecture

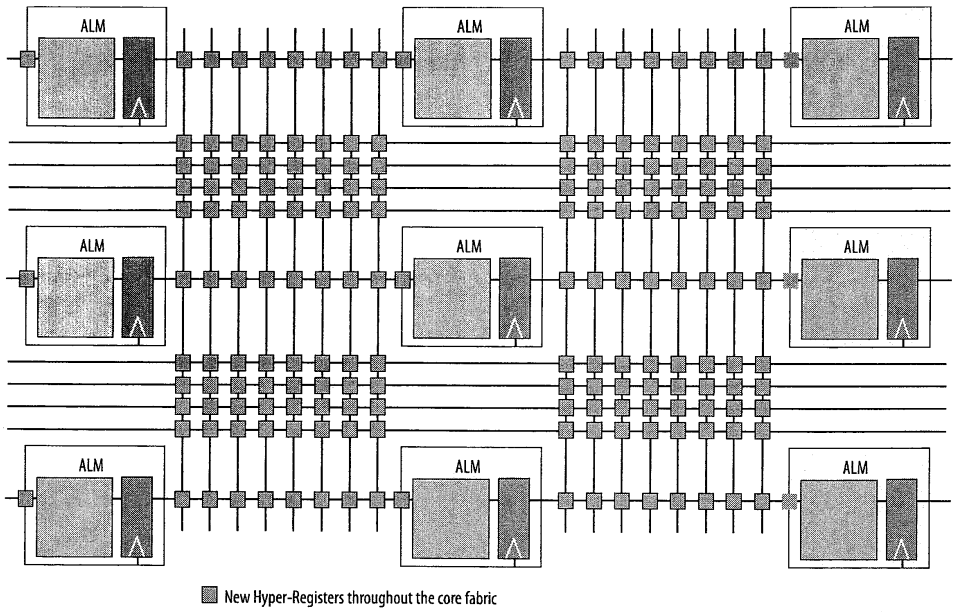
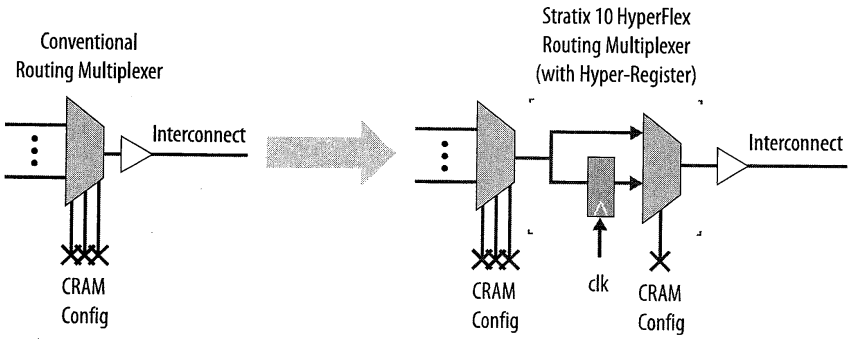


Figure 2: Bypassable Hyper-Register

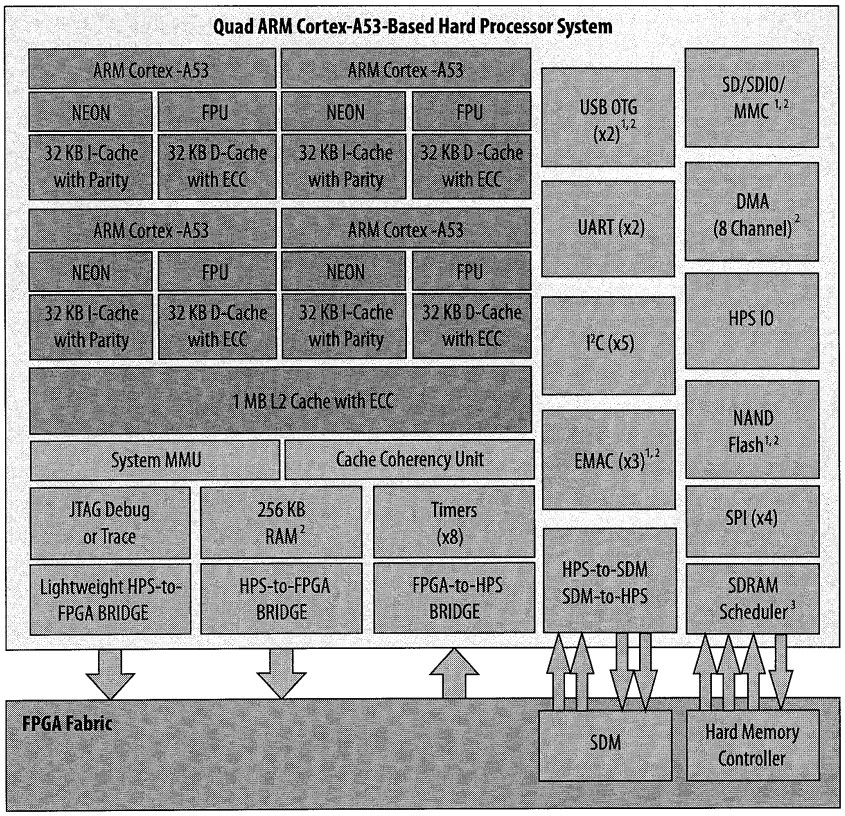


The Hyper-Registers enable the following key design techniques to achieve the 2X core performance increases:

- Fine grain Hyper-Retiming to eliminate critical paths
- Zero latency Hyper-Pipelining to eliminate routing delays
- Flexible Hyper-Optimization for best-in-class performance

By implementing these techniques in your design, the Hyper-Aware design tools automatically make use of the Hyper-Registers to achieve maximum core clock frequency.

Figure 13. HPS Block Diagram



Notes:
1. Integrated direct memory access (DMA)
2. Integrated error correction code (ECC)
3. Multiport front-end interface to hard memory controller

1.18.1 Key Features of the Intel Stratix 10 HPS

Table 14. Key Features of the Intel Stratix 10 GX/SX HPS

Feature	Description
Quad-core ARM Cortex-A53 MPCore processor unit	• 2.3 MIPS/MHz instruction efficiency • CPU frequency up to 1.5 GHz • At 1.5 GHz total performance of 13,800 MIPS • ARMv8-A architecture • Runs 64-bit and 32-bit ARM instructions • 16-bit and 32-bit Thumb instructions for 30% reduction in memory footprint • Jazelle® RCT execution architecture with 8-bit Java bytecodes

continued...

The DSP blocks can be configured to support signal processing with precision ranging from 18x19 up to 54x54. A pipeline register has been added to increase the maximum operating frequency of the DSP block and reduce power consumption.

Figure 10. DSP Block: Standard Precision Fixed Point Mode

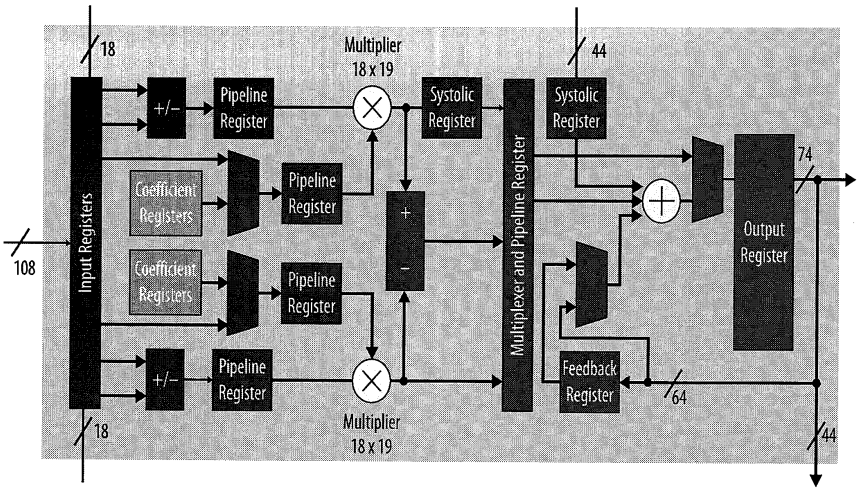
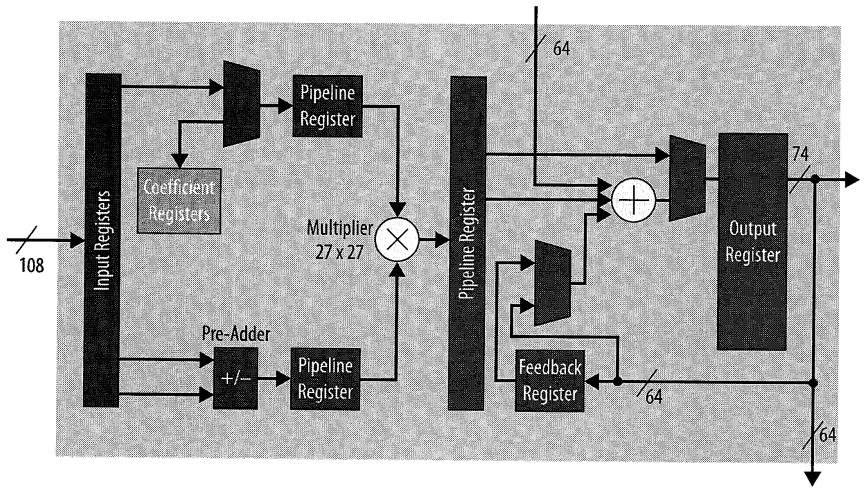


Figure 11. DSP Block: High Precision Fixed Point Mode



INTEL® STRATIX® 10 PRODUCT TABLE

Stratix® 10

	PRODUCT LINE	GX 400 SX 400	GX 650 SX 650	GX 850 SX 850	GX 1100 SX 1100	GX 1650 SX 1650	GX 2100 SX 2100	GX 2500 SX 2500	GX 2800 SX 2800	GX 4500 SX 4500	GX 5500 SX 5500
Resources	Logic elements (LEs)[1]	378,000	612,000	841,000	1,092,000	1,624,000	2,005,000	2,422,000	2,753,000	4,463,000	5,510,000
	Adaptive logic modules (ALMs)	128,160	207,360	284,960	370,080	550,540	679,680	821,150	933,120	1,512,820	1,867,680
	ALM registers	512,640	829,440	1,139,840	1,480,320	2,202,160	2,718,720	3,284,600	3,732,480	6,051,280	7,470,720
	Hyper-Registers from Intel® HyperFlex™ FPGA architecture	Millions of Hyper-Registers distributed throughout the monolithic FPGA fabric									
	Programmable clock trees synthesizable	Hundreds of synthesizable clock trees									
	M20K memory blocks	1,537	2,489	3,477	4,401	5,851	6,501	9,963	11,721	7,033	7,033
	M20K memory size (Mb)	30	49	68	86	114	127	195	229	137	137
	MLAB memory size (Mb)	2	3	4	6	8	11	13	15	23	29
	Variable-precision digital signal processing (DSP) blocks	648	1,152	2,016	2,520	3,145	3,744	5,011	5,760	1,980	1,980
	18 x 19 multipliers	1,296	2,304	4,032	5,040	6,290	7,488	10,022	11,520	3,960	3,960
	Peak fixed-point performance (TMACS)[2]	2.6	4.6	8.1	10.1	12.6	15.0	20.0	23.0	7.9	7.9
	Peak floating-point performance (TFLOPS)[3]	1.0	1.8	3.2	4.0	5.0	6.0	8.0	9.2	3.2	3.2
I/O and Architectural Features	Secure device manager	AES-256/SHA-256 bitsream encryption/authentication, physically unclonable function (PUF), ECDSA 256/384 boot code authentication, side channel attack protection									
	Hard processor system[4]	Quad-core 64 bit ARM* Cortex*-A53 up to 1.5 GHz with 32 KB I/D cache, NEON* coprocessor, 1 MB L2 cache, direct memory access (DMA), system memory management unit, cache coherency unit, hard memory controllers, USB 2.0 x2, 1G EMAC x3, UART x2, SPI x4, I²C x5, general-purpose timers x7, watchdog timer x4									
	Maximum user I/O pins	392	400	736	736	704	704	1160	1160	1640	1640
	Maximum LVDS pairs 1.6 Gbps (RX or TX)	192	192	360	360	336	336	576	576	816	816
	Total full duplex transceiver count	24	48	48	48	96	96	96	96	24	24
	GXT full duplex transceiver count (up to 30 Gbps)	16	32	32	32	64	64	64	64	16	16
	GX full duplex transceiver count (up to 17.4 Gbps)	8	16	16	16	32	32	32	32	8	8
	PCI Express* (PCIe*) hard intellectual property (IP) blocks (Gen3 x16)	1	2	2	2	4	4	4	4	1	1
	Memory devices supported	DDR4, DDR3, DDR2, DDR, QDR II, QDR II+, RLDRAM II, RLDRAM 3, HMC, MoSys									

Package Options and I/O Pins: General-Purpose I/O (GPIO) Count, High-Voltage I/O Count, LVDS Pairs, and Transceiver Count[5]

F1152 pin (35 mm x 35 mm, 1.0 mm pitch)	392,8,192,24	392,8,192,24	–	–	–	–	–	–	–	–
F1760 pin (42.5 mm x 42.5 mm, 1.0 mm pitch)	–	400,16,192,48	–	–	–	–	–	–	–	–
F1760 pin (42.5 mm x 42.5 mm, 1.0 mm pitch)	–	–	688,16,336,48	688,16,336,48	688,16,336,48	688,16,336,48	688,16,336,48	688,16,336,48	–	–
F2112 pin (47.5 mm x 47.5 mm, 1.0 mm pitch)	–	–	736,16,360,48	736,16,360,48	–	–	–	–	–	–
F2397 pin (50 mm x 50 mm, 1.0 mm pitch)	–	–	–	–	704,32,336,96	704,32,336,96	704,32,336,96	704,32,336,96	–	–
F2912 pin (55 mm x 55 mm, 1.0 mm pitch)	–	–	–	–	–	–	1160,8,576,24	1160,8,576,24	1640,8,816,24	1640,8,816,24

Notes:
1. LE counts valid in comparing across Intel FPGA devices, and are conservative vs. competing FPGAs.
2. Fixed-point performance assumes the use of pre-adder.
3. Floating-point performance is IEEE 754 compliant single precision.
4. Quad-core ARM Cortex-A53 hard processor system only available in Stratix 10 SX SoCs.
5. A subset of pins for each package are used for high-voltage, 3.0 V and 2.5 V interfaces.
6. All data is preliminary, and may be subject to change without prior notice.

392,8,192,24 Numbers indicate total GPIO count, high-voltage I/O count, LVDS pairs, and transceiver count.

Indicates pin migration path.

INTEL STRATIX 10 PRODUCT TABLE

Stratix® 10

PRODUCT LINE	SX 400	SX 650	SX 850	SX 1100	SX 1650	SX 2100	SX 2500	SX 2800	SX 4500	SX 5500
Processor	Quad-core 64 bit ARM Cortex-A53 MPCore* processor									
Maximum processor frequency	1.5 GHz[1]									
Processor cache and co-processors	• L1 instruction cache (32 KB) • L1 data cache (32 KB) with error correction code (ECC) • Level 2 cache (1 MB) with ECC • Floating-point unit (FPU) single and double precision • ARM NEON media engine • ARM CoreSight* debug and trace technology • System Memory Management Unit (SMMU) • Cache Coherency Unit (CCU)									
Scratch pad RAM	256 KB									
HPS DDR memory	DDR4, DDR3, and LP DDR3 (Up to 64 bit with ECC)									
Direct memory access (DMA) controller	8 channels									
EMAC	3X 10/100/1000 Ethernet media access controller (EMAC) with integrated DMA									
USB on-the-go (OTG) controller	2X USB OTG with integrated DMA									
UART controller	2X UART 16550 compatible									
Serial peripheral interface (SPI) controller	4X SPI									
I²C controller	5X I²C									
Quad SPI flash controller	1X SIO, DIO, QIO SPI flash supported									
SD/SDIO/MMC controller	1X eMMC 4.5 with DMA and CE-ATA support									
NAND flash controller	• 1X ONFI 1.0 or later • 8 and 16 bit support									
General-purpose timers	4X									
Software-programmable general-purpose I/Os (GPIOs)	Maximum 48 GPIOs									
HPS DDR Shared I/O	3X 48 - May be assigned to HPS for HPS DDR access									
Direct I/Os	48 I/Os to connect HPS peripherals directly to I/O									
Watchdog timers	4X									
Security	Secure device manager, Advanced Encryption Standard (AES) AES-256/SHA-256 bitsream encryption/authentication, PUF, ECDSA 256/384 boot code authentication, side channel attack protection									

Notes:
1. With overdrive feature.

Gen-1023-1.5

ASIC 22/11/2017

Valutazioni sul flusso progettuale.

Gli ASIC ci portano ad un incremento di prestazioni in ogni direzione:

- velocità di elaborazione;
- area;
- potenza;
- sicurezza del progetto;
- affidabilità;

Tutto questo viene pagato in termini di costo $.

In un ASIC si progetta sia l'architettura logica che il layout -> il design è più difficile e comporta l'aumento del tempo di progetto.

Si richiedono strumenti di CAD avanzati.

È inoltre necessario effettuare un collaudo del chip progettato.

La produzione del chip sarà inoltre off-house (sarà eseguito da aziende specializzate).

Approccio Full custom:

Libertà progettuale totale e completa
-> ottimizzazione geometrica ed elettrica a livello di transistor.

Approccio Semi custom:

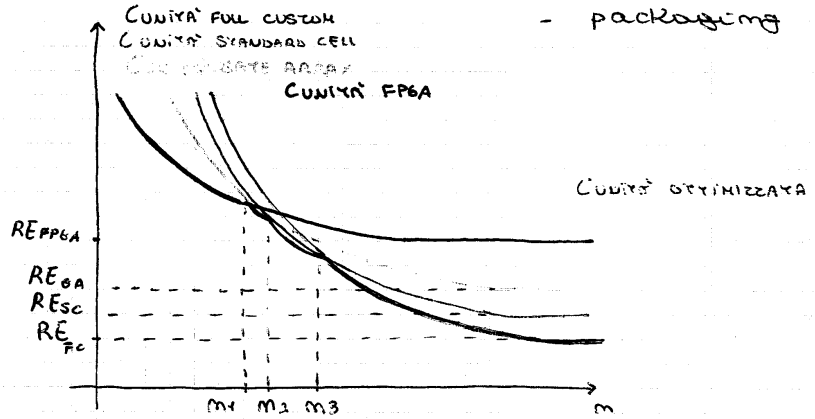
- utilizzo di Standard Cell (celle progettuali pre-progettate);
- Gate array channelled o Sea of Gate;
  -> I transistori sono già prefabbricati sul chip, si personalizzano solo le connessioni.

$$Costo_{TOT} = NRE + n \cdot RE$$

$$Costo_{TOT}/n = NRE/n + RE = Costo_{UNIT.}$$

Costi fissi NRE:
- maschere di progetto
- progettisti

Costi ricorrenti RE
- area di silicio
- collaudo
- packaging



$RE_{FULL\ CUSTOM} < RE_{STANDAR\ CELL} < RE_{GATE\ ARRAY}$

$NRE_{FULL\ CUSTOM} > NRE_{STANDAR\ CELL} > NRE_{GATE\ ARRAY}$

Sicuramente:

$RE_{FPGA} > RE_{GATE\ ARRAY}$

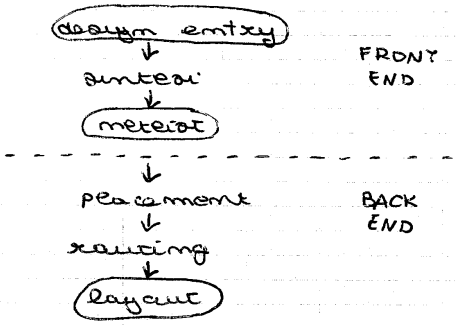
$NRE_{FPGA} < NRE_{GATE\ ARRAY}$

La prototipazione è fatta su FPGA (architetture programmabile). Al termine del progetto si migra su una tecnologia gate array o standard cell.

-> Prototipazione veloce e implementazione ottimizzata.

Flusso progettuale ASIC digitale:

1) design entry;
2) sintesi in primitive;
3) trasferimento su silicio:
   - placement;
   - routing;



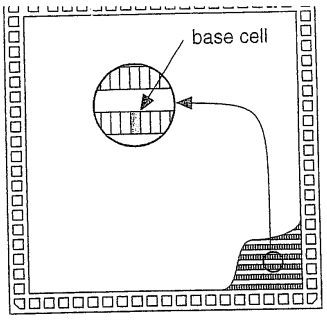


FIGURE 1.5 A channeled gate-array die. The spaces between rows of the base cells are set aside for interconnect.

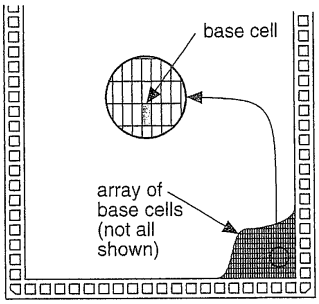
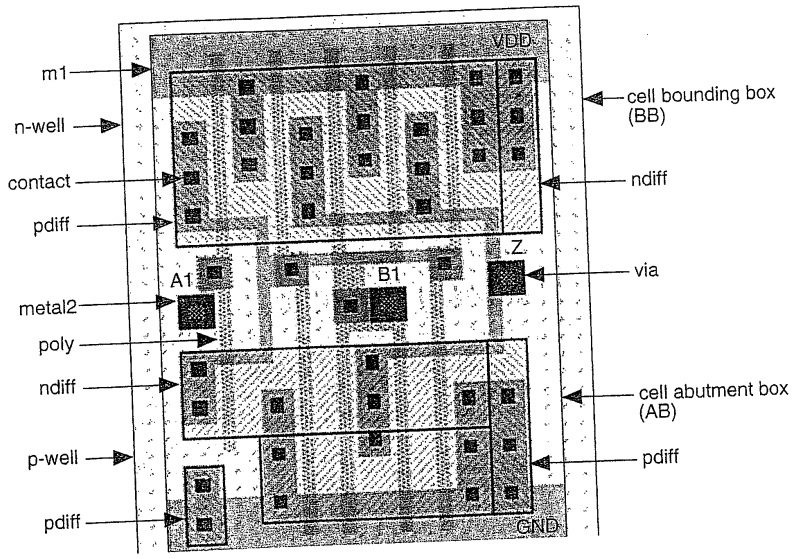
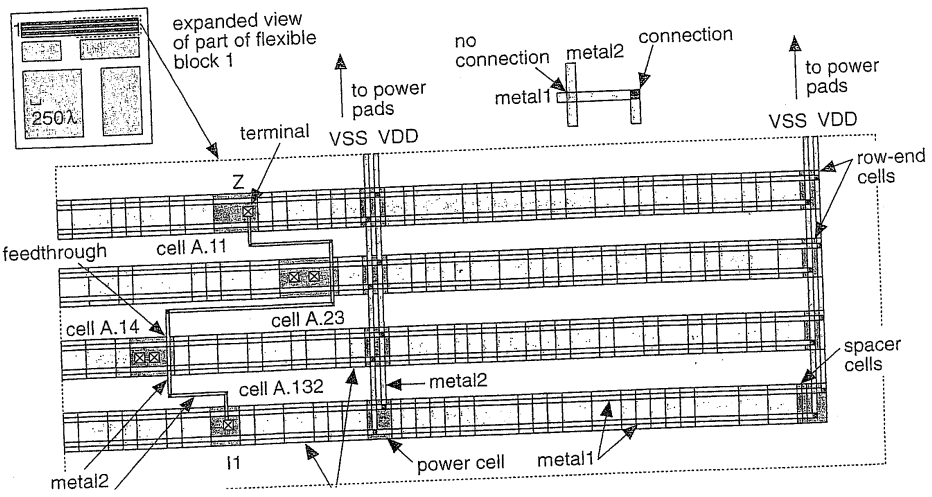


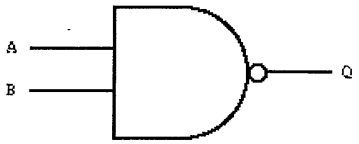
FIGURE 1.6 A channelless gate-array or sea-of-gates (SOG) array die. The core area of the die is completely filled with an array of base cells (the base array).





Conditions for characterization library **c18_CORELIB_TYP**, corner **c18_CORELIB_TYP_typical**: Vdd= **1.80**V, Tj= **25.0** deg. C .
Output transition is defined from **20%** to **80%** (rising) and from **80%** to **20%** (falling) output voltage.
Propagation delay is measured from **50%** (input rise) or **50%** (input fall) to **50%** (output rise) or **50%** (output fall).

Strength	1
Cell Area	11.290 um^2
Equation	Q = "!(A & B)"
Type	Combinational
Input	A, B
Output	Q



State Table		
A	B	Q
L	-	H
H	H	L
-	L	H

transizione veloce

transizione lenta

Propagation Delay [ns]					
Input Transition [ns]		0.01		2.50	
Load Capacitance [fF]		1.00	100.00	1.00	100.00
A to Q	fall	0.03	0.61	-0.08	1.21
	rise	0.04	1.04	0.41	1.93
B to Q	fall	0.03	0.62	-0.11	1.07
	rise	0.04	1.05	0.47	1.94

Output Transition [ns]					
Input Transition [ns]		0.01		2.50	
Load Capacitance [fF]		1.00	100.00	1.00	100.00
A to Q	fall	0.02	0.80	0.29	1.25
	rise	0.04	1.48	0.28	1.75
B to Q	fall	0.02	0.80	0.27	1.12
	rise	0.05	1.49	0.27	1.76

carico presente

carico contenuto.

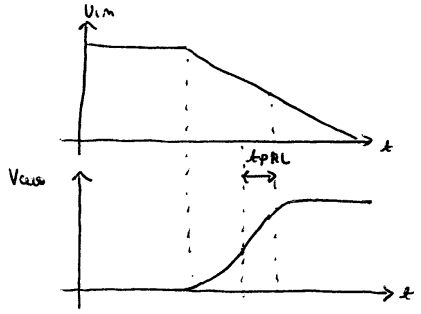
Capacitance [fF]	
A	1.5400
B	1.4830

Leakage [pW]
22.97

Dynamic Power Consumption [nW/MHz]					
Input Transition [ns]		0.01		2.50	
Load Capacitance [fF]		1.00	100.00	1.00	100.00
A to Q	fall	0.00	0.37	19.54	10.52
	rise	6.65	6.98	33.12	20.86
B to Q	fall	0.00	0.21	18.15	10.26
	rise	8.14	8.37	35.99	22.98

Tempo Propagazione negativo

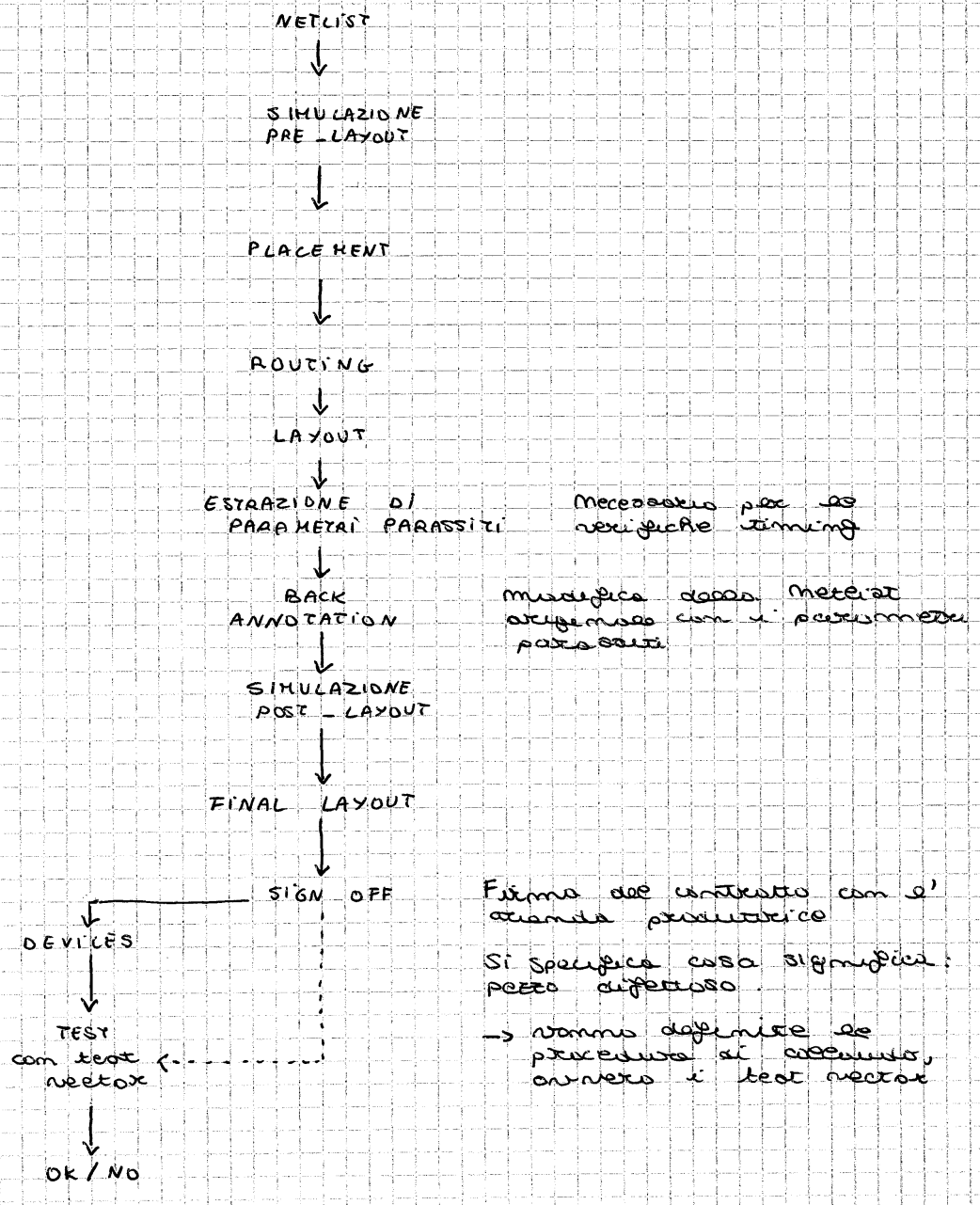
Transizione di input lenta:



Il ritardo della temporizzazione di uscita tp risulta minore di ϕ per transizione lenta di Vin

24/11/2017

Se progetto ASIC richiede di funzionare al primo RUN al silicio -> non è contemplato a causa dei costi NRE di commettere errori.



τ : ritardo di propagazione di una porta elementare

$$\tau = \tau_0 + K \sum_{i=1}^{N} C_{in_i}$$

τ è funzione di:

- temperatura;
- tensione di alimentazione;
- processo.

Da sottolineare che $\tau_{HL} \neq \tau_{LH}$:

$$\tau_{HL} = \tau_{0HL} + K_{HL} \sum_{i=1}^{N} C_{in_i}$$

$$\tau_{LH} = \tau_{0LH} + K_{LH} \sum_{i=1}^{N} C_{in_i}$$

Si determinano τ per per formance:

- slow (T_{MAX}, $V_{ALIM\,MIN}$, PROCESSO PESSIMO)
- typical
- fast (T_{MIN}, $V_{ALIM\,MAX}$, PROCESSO OTTIMO)

Post Layout si introduce in τ un termine C_R dovuta al routing:

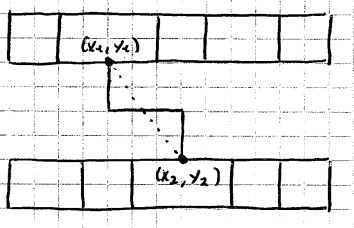
$$\tau = \tau_0 + K\left[\sum_{i=1}^{N} C_{in_i} + C_R\right]$$

PLACEMENT

Posizionamento intelligente delle standard cell.

Criterio di stima:

- minimizzare C_R: minimizzare il ROUTING.



$D = |x_1 - x_2| + |y_1 - y_2|$

Distanza P_1, P_2.

Si utilizza un algoritmo di Random Placement iterativo.

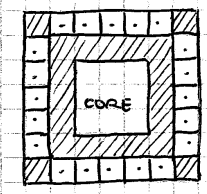
ROUTING

Si divide in global routing e Detailed Routing.

BONDING DIAGRAM

Schemma di collegamento PIN out chip con PIN out Package.

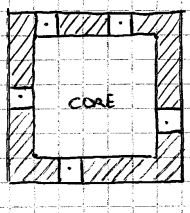
Risultato finale:



: silicio libero.

Chip Pad limited:

-> dimensione chip legata dal numero di PAD.



Chip Core limited

-> dimensione chip legata dalla dimensione del core.

METODOLOGIE DI PROGETTO

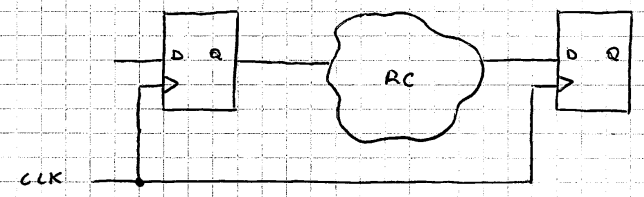
Una rete digitale è una grande rete sequenziale.

Tipi di progetto:

1) Progetto Sincrono Statico

Gli elementi di memoria FF sono elementi statici che mantengono informazione anche in assenza di clock.

Tutti i FF sono clockati dallo stesso fronte dello stesso segnale di clock.



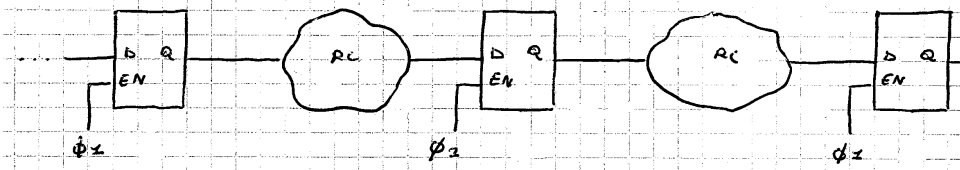
2) Progetto multifase Statico

FF statici attivati da clock multifase.

Non si utilizzano i FF ma i LATCH.

Nel caso abbastanza si usano due fasi: ϕ_1, ϕ_2.

Segnali elaborati su ϕ_1 devono essere ricevuti su ϕ_2.



3) Progetto multifase Dinamico

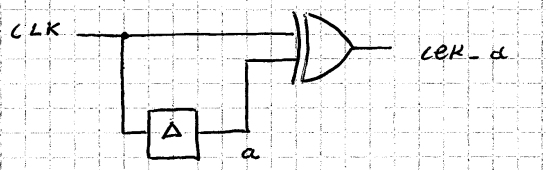
I latch che si utilizzano sono di tipo dinamico.

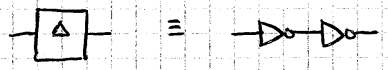
-> la memorizzazione avviene sulla C in di un LATCH.

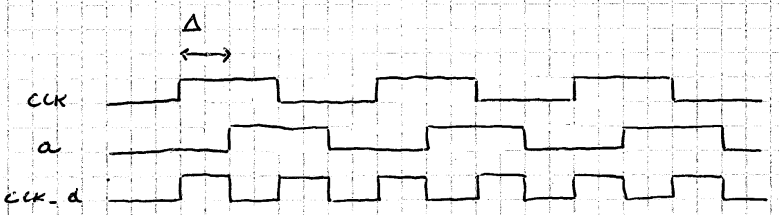
INTRODUZIONE DI UN RITARDO CON PORTE IN CASCATA

Linee di ritardo

Moltiplicatore di frequenza





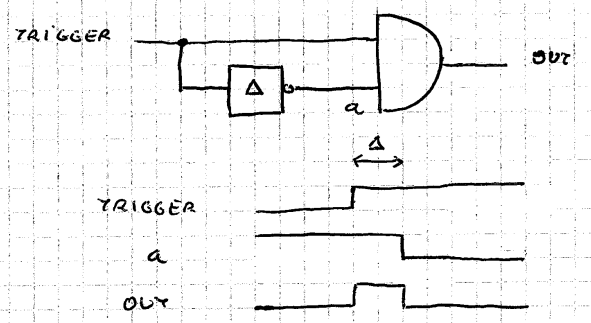


$DC_{CLK} = 50\,\%$

$\Delta = \frac{T_{CLK}}{4} \rightarrow DC_{CLK\_d} = 50\,\%$

Δ è però un parametro estremamente poco preciso in quanto è legato ai t_{PHL} e t_{PLH}.

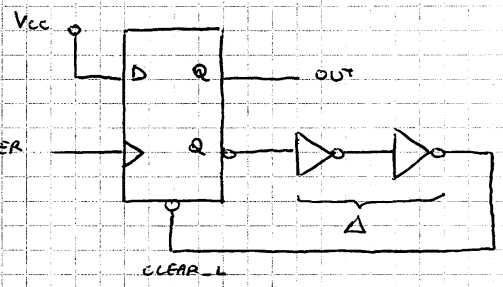
Altra soluzione: usare un monostabile digitale.

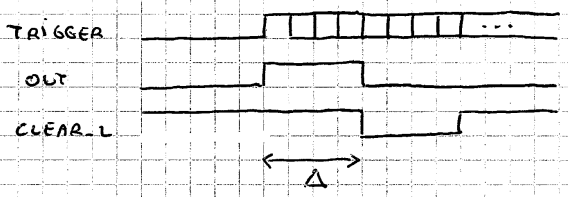


Δ effetto dei soliti problemi.

Trigger deve essere valido per tutta la transizione.

Vediamo il comportamento del monostabile visto:

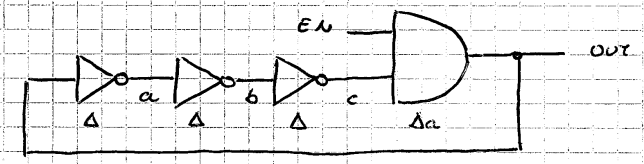


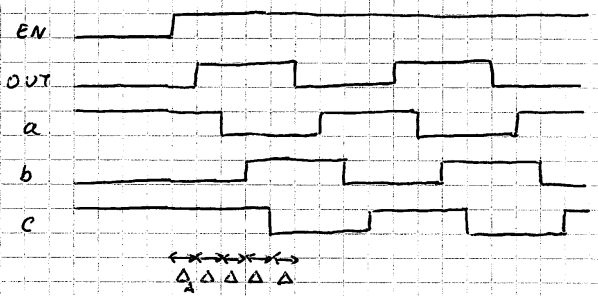


Basta le fronte di trigger per settare il FF.

Ancora il problema della non esattezza di Δ.

ASTABILE



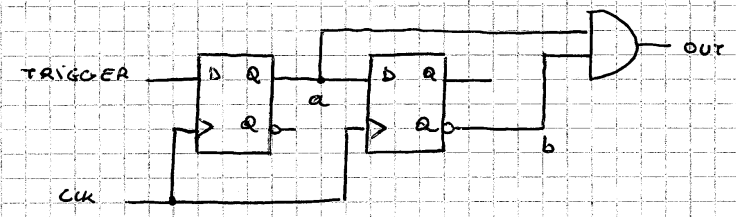


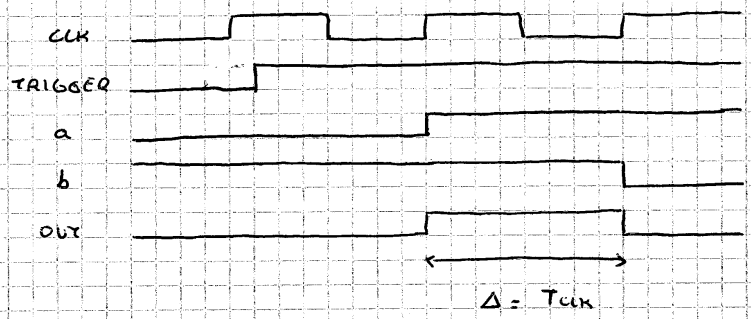
Problema della inesattezza di Δ.

Il clock è un elemento che dà un riferimento temporale.

-> il periodo del clock consente di misurare porzioni di tempo.

Uso come Δ un intervallo di clock:





Δ estremamente preciso

Da notare che l'attivazione del Trigger si riconduce ovvio su $\uparrow$.

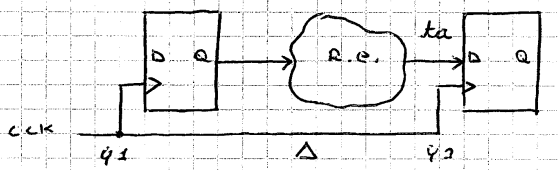
T_{CLK} fornisce la risoluzione del sistema.

29/11/2017

Se vogliamo misurare intervalli temporali è necessario esprimerli sul periodo di clock.

Criteri di scelta del clock per sistemi digitali:

- periodo di clock corrisponde al più piccolo quanto temporale che è necessario misurare.
  -> esprimo tutti gli altri intervalli temporali come multipli del periodo di clock.
- il periodo di clock definisce l'intervallo di campionamento di un segnale -> non è possibile discriminare un segnale all'interno di tale intervallo.
- Ci poniamo sempre riferiti ad una struttura in cui le prestazioni sono determinate dal cammino critico.



$$\begin{cases} T_{CLK} > t_{CO} + t_{PD} + t_{SU} \\ t_{CO} + t_{PD} > t_H \end{cases}$$

Vero se il clock è esente da sbilanciamenti.

Ora presenza di sbilanciamenti:

- CLOCK SKEW •

$\phi_2 = \phi_1 \pm \Delta$ con $\Delta > 0$

$t_a = \phi_1 + t_{CO} + t_{PD} \leq \phi_1 \pm \Delta + T_{CLK} - t_{SU}$

$\pm \Delta \geq t_{CO} + t_{PD} + t_{SU} - T_{CLK} < 0$

$+\Delta$: skew non arreca problemi

$-\Delta$: $|\Delta| \leq T_{CLK} - t_{CO} - t_{PD} - t_{SU}$

Timer Analyzer chiama $|\Delta|$ SLACK : è il clock skew tollerabile.

$t_a = 4_x + t_{CO} + t_{PD}$

$t_a > 4_2 + t_H = 4_x \pm \Delta + t_H$

Caso peggiore: $t_{PD} = \not{0}$ ($R, e = c.c.$)

$4_x + t_{CO} > 4_x \pm \Delta + t_H$

$\pm \Delta < t_{CO} - t_H > \not{0}$

$-\Delta$: skew non crea problemi

$+\Delta$: $|\Delta| < t_{CO} - t_H$

Primitive asincrone:

FF - SR asincrono

- $S=1$, $R=1$ configurazione ignota
- asincrono in un mondo sincrono
- sensibile ai glitch
- nel caso di set $S=1$, resettando ($R=1$) mi porto in configurazione ignota.

FF - JK

Consente una scrittura più difficoltosa dello schema circuitale.

01/12/2017

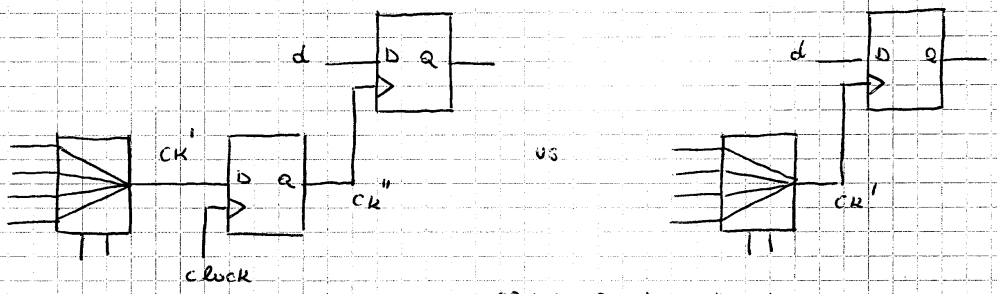
Reti combinatorie che pilotano reti di clock e reset asincroni:

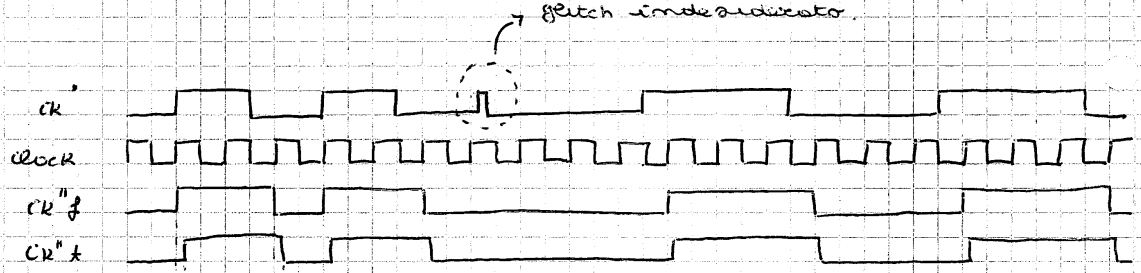
Una rete combinatoria può determinare delle criticità a causa delle ALEE (transitori spurie). Ci possono essere transitori spurie delle uscite che sono indesiderate → la funzionalità del sistema varia.

Da un'analisi funzionale questo però non risulta; è necessario un'analisi approfondita timing.

nel caso in cui si abbia un'architettura completamente clock multiplo o una rete combinatoria per controllare un reset asincrono inserisco un registro.

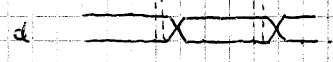
Rischio il problema della presenza di glitch sul clk.





CK"f: funziona

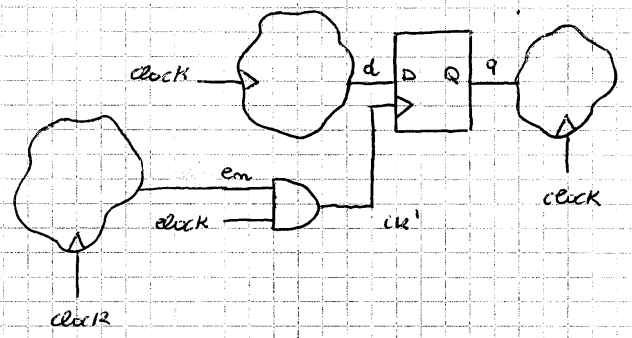
CK"t: scalo (timing) con presenza di clock skew.



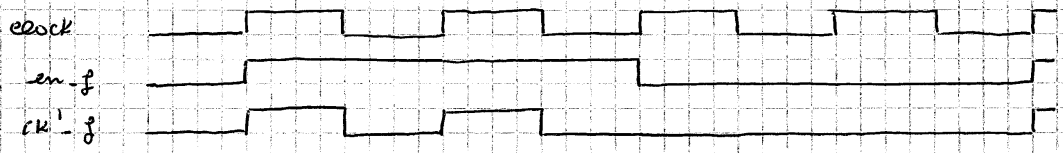
potremmo notare che th per CK" = CK"_t

-> lo skew positivo peggiora le prestazioni nel rispetto di th. Si genera dunque un segnale ripulito ma passibile di clock skew.

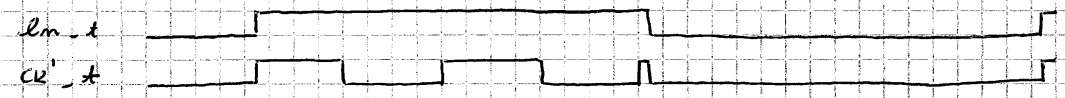
TECNICA DEL CLOCK GATING



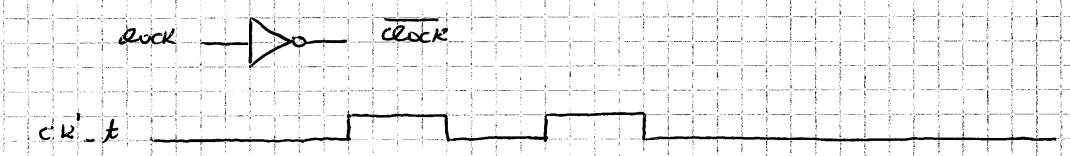
Disabilitazione della macchina tramite disabilitazione del clock.



In realtà, poi come è generato en, esso avverrà in ritardo rispetto a clock.



Se inverto invece che clock → $\overline{clock}$



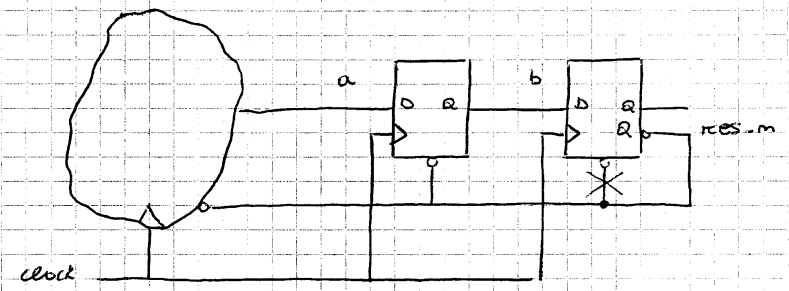
Non quindi il fronte negativo del clock evitando così i problemi introdotti dal clock gating.

Ma q viene generato sul ↑ di ck'_t e viene complementato su ↑ di clock, cioè ↓ di ck'_t.

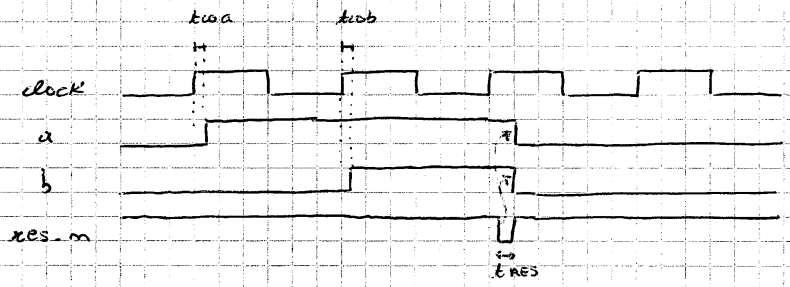
→ abbiamo solo mezzo periodo di clock per mandare a regime l'ingresso q. (se $DC_{clock} = 50\%$)

È una struttura DOUBLE EDGE CLOCKING.

RESET ASINCRONO

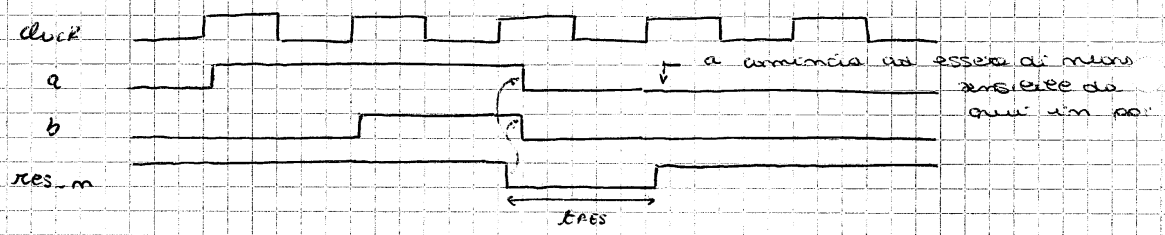


Struttura che resetta nel momento in cui res_n = 0



Un impulso di reset così di breve durata è possibile che non sia in grado di resettare tutti i flip flop che vogliamo resettare.

Provo a rimediare questo collegamento

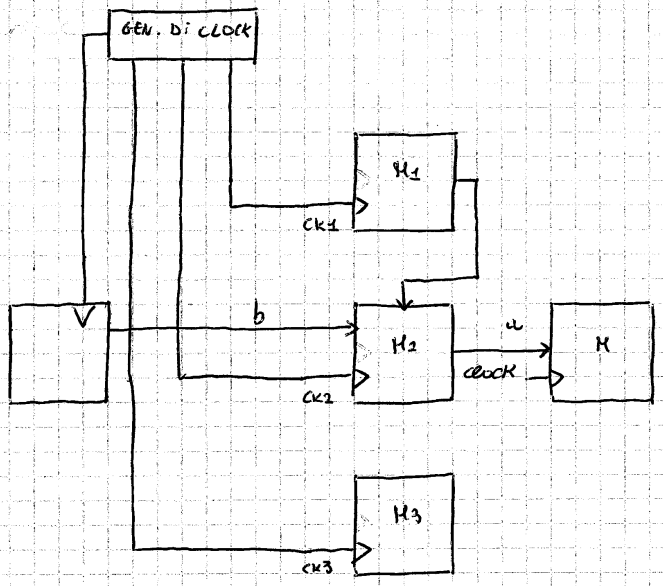


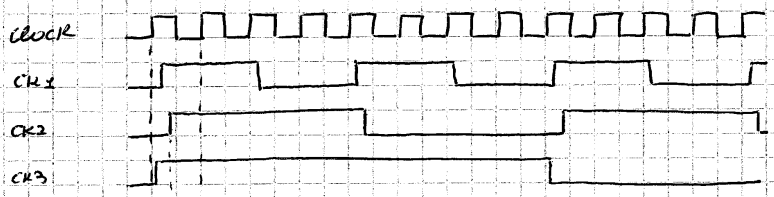
Così Ho costruito un impulso di reset di lunga durata.

In questo caso l'ultimo FF serve esclusivamente da FF di reset.

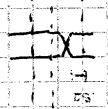
Problema: perdo un ciclo di clock -> si inserisce un ciclo morto di clock.

MACCHINE CLOCKATE CON CLOCK DIVERSI





C'è uno sfasamento nella generazione del clock.

a 

a generato da ck2 ma campionato da clock

-> si potrebbero violare tempi di set up.

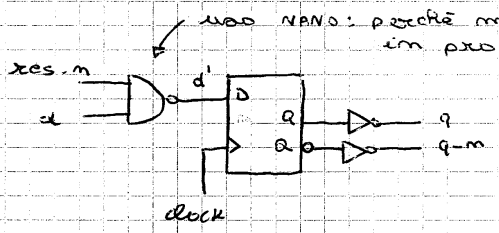
b 

b generato da clock ma campionato da ck2

-> si potrebbero violare tempi di hold.

FF D con reset sincrono è la struttura base ideale per il progetto sincrono.

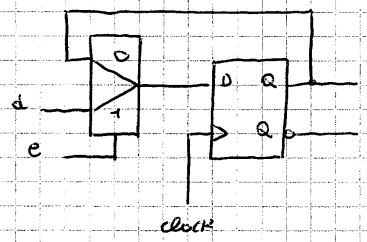
Costruire un reset sincrono se non ho la cella sincrona in libreria:



$d' = \overline{res\_n \cdot d}$

-> struttura che non vede a regime dal progetto sincrono statico.

FF E

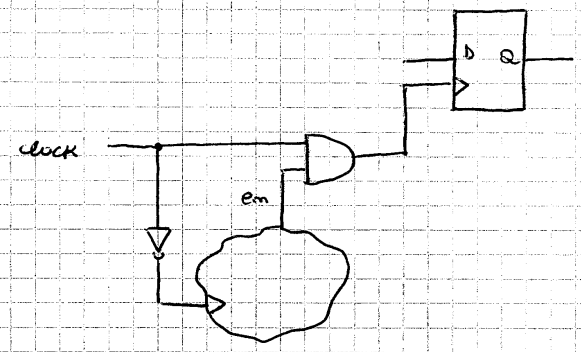


Disabilitazione dell'ingresso in funzione di e.

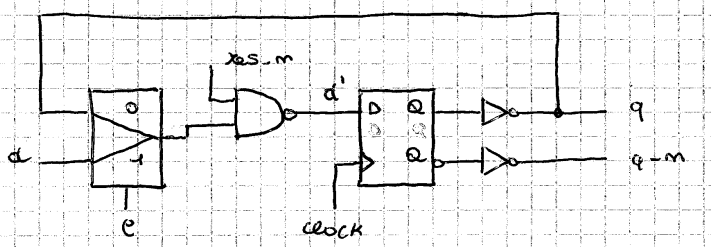
-> non è clock gating ma FLIP FLOP con abilitazione.

Il campionamento avviene per ogni fronte di clock due fronte ↑.

Il CLOCK GATING viene utilizzato quasi esclusivamente per problemi di consumo energetico.



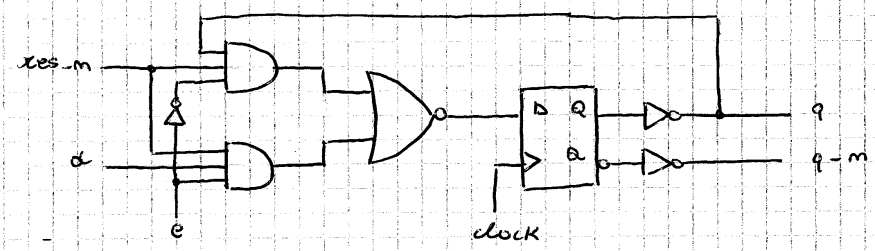
FF E con reset sincrono:



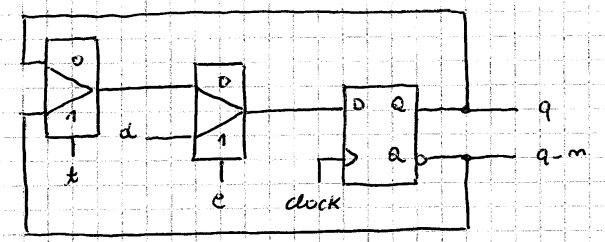
AND_OR_INVERT (AOI)

$$d' = \overline{res\_m \cdot (d \cdot e + q \cdot \bar{e})} = \overline{res\_m \cdot d \cdot e + res\_m \cdot q \cdot \bar{e}}$$

Si traduce in:



FF T - T



→ abilitabile sia come campionatore che come contatore.

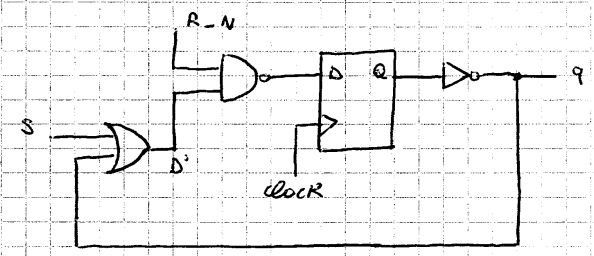
Si usano nei contatori così cablati.

$e = 1$ -> si campiona sincronamente d

$e = 0$, $t = 0$ -> mantenimento

$e = 0$, $t = 1$ -> conteggio

FF $S\bar{R}$



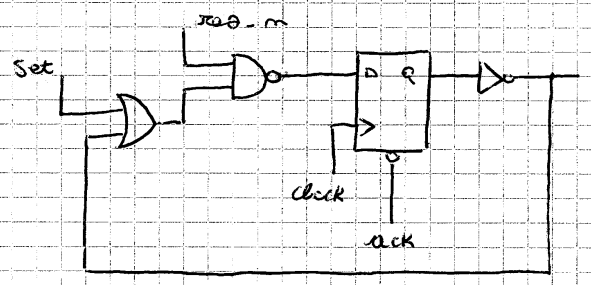
$D' = S + Q$

$S = 1$	$R\_N = 1$	$Q \Leftarrow 1$
$S = \emptyset$	$R\_N = 1$	$Q \leftarrow Q$
$S = X$	$R\_N = \emptyset$	$Q \Leftarrow \emptyset$

Risolve i problemi del latch SR:

- anisteresi
- campionamento sui fronti
- genera uscite sui fronti
- risolve problema $S = R = 1$

FF R (CRequest FF)



05/12/2017

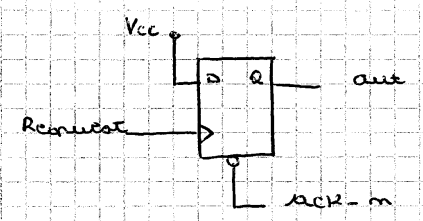
Ha un reset asincrono pilotato da ack.

Utilizziamo questo FF alla periferia del sistema, quando vogliamo lanciare delle richieste verso un altro sistema aspettiamo un acknowledgment. Serve per fare comunicare il sistema digitale col mondo esterno.

Reset asincrono in quanto il mondo esterno agisce in maniera asincrona.

FF - R per gestione richieste esterne (dall'esterno)

→ FF non clockato da CLK di sistema con reset asincrono



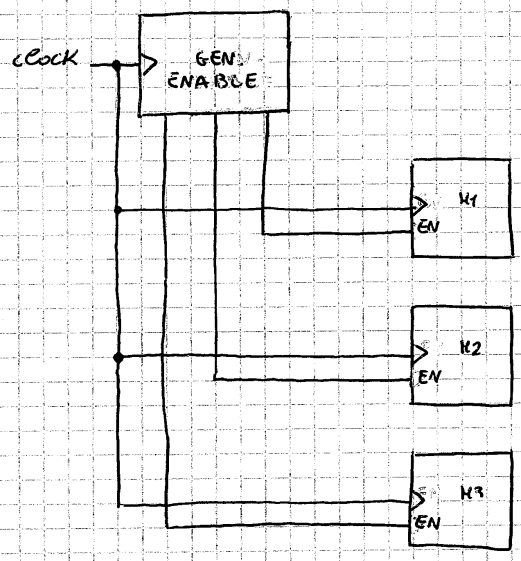
Questo oggetto si trova nella periferia del sistema.

Può servire per aggiungere un segnale proveniente dal mondo esterno, collegata la richiesta di reset.

Questo non FF che vedremo in regole di progetto ma usato per interfacciarsi col mondo esterno.

Ottenere delle macchine sequenziali ad intervalli diversi.

→ Risolve le problematiche di macchine che funzionano con periodi di clock differenti.



Il progetto è totalmente sincrono statico evoluto.

→ non vale le regole di progetto viste.

Regole per il progetto digitale:

1) Progetto funzionale:
   scrivere del codice ordinato facile da comprendere, verificare e correggere.

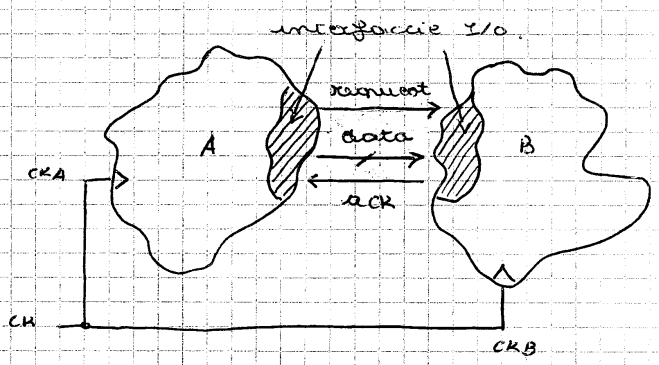
2) Evitare le tecniche sconsigliate.

3) Il progetto digitale funziona bene e tutti i FF campionano correttamente (rispetto t_{SU} e t_H dei segnali)
   → internamente alla critical path va rispettata $f_{CLK\,MAX}$; dei segnali esterni bisogna rispettare t_{SU} e t_H.

N.B: segnali esterni tipicamente sincroni rispetto a CLK sono segnali che vedono t_H e t_{SU} → di più non controllano la situazione

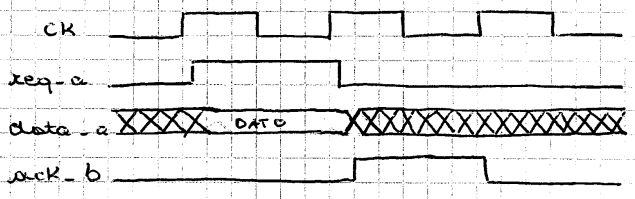
Il terzo punto si traduce in: Gestione della comunicazione con l'esterno.

possiamo solo controllare cosa accade mediante i vincoli di relazione di t_{SU} e t_H.



I moduli sono sincroni.

$CK = CKA = CKB$



Supponiamo che i moduli non siano sincroni:

$CKA \neq CKB$

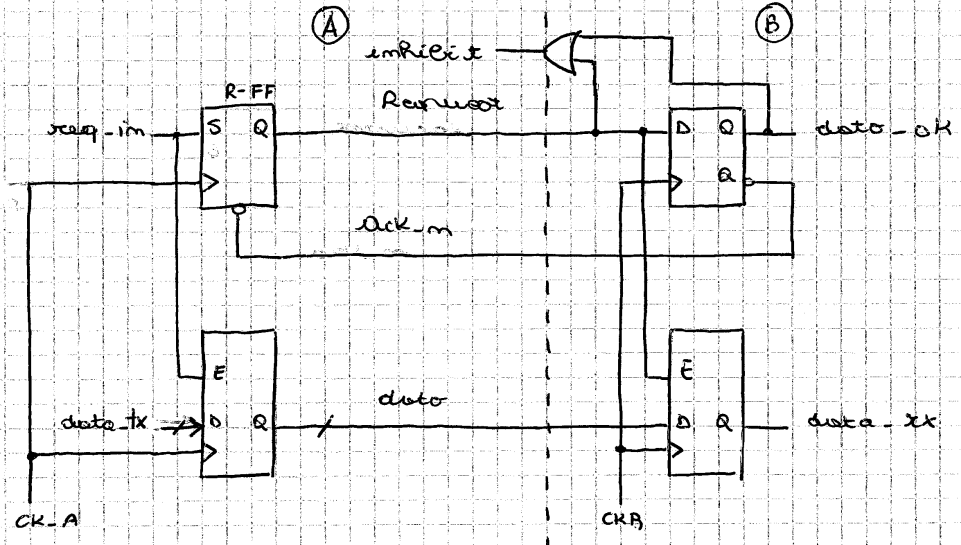
Se da A fluiscono un dato per ogni ciclo di clock CKA, deve risultare:

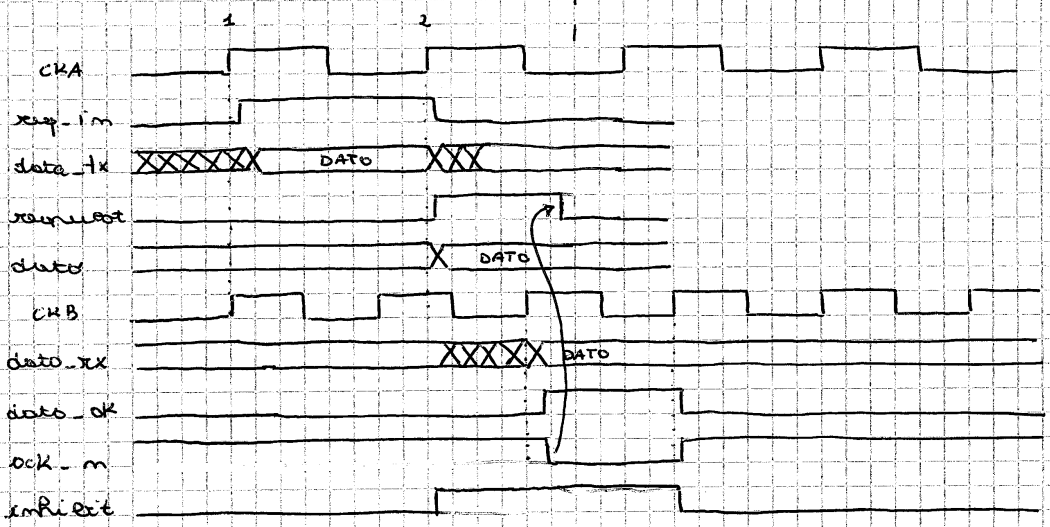
$f_{CKB} \geq f_{CKA}$.

È necessario un protocollo di comunicazione tra A e B.

Si usa un HandShake.

Protocollo di HandShake Fondamentale:





inhibit indica l'occupazione della linea di comunicazione.

Abbiamo studiato un caso fortuito in cui i dati provenienti da A non ricadono tra t_s e t_H del sistema B.

Potrebbe accadere che Request nei tempi t_H e t_{SU} del FF di sincr. di tipo D.

Il FF di tipo D viene retto dall'ingresso durante la metastabilità. Supponiamo che Request non venga campionato e Q = ∅ -> questo non è un problema: non c'è riconoscimento della richiesta, quindi non c'è attraversamento e la richiesta rimane attiva. Sul fronte successivo di CK B si campionamento è corretto. Supponiamo che Request venga campionato e Q = 1 -> dobbiamo considerare che anche dato vale i tempi t_H e t_{SU} e quindi non è assicurato il corretto campionamento.

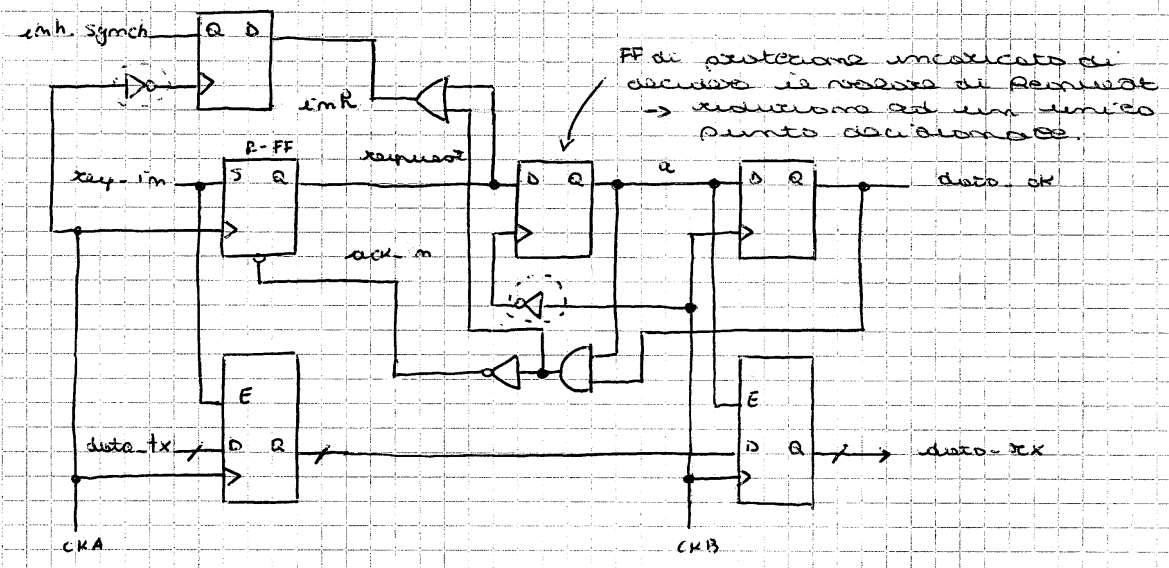
C'è una probabilità di errore!

Il problema nasce dal fatto che Request è asincrono rispetto a CK B. Inoltre ci sono due punti decisionali: interni all'interno del sistema B in cui si verifica la validità del segnale asincrono in ingresso.

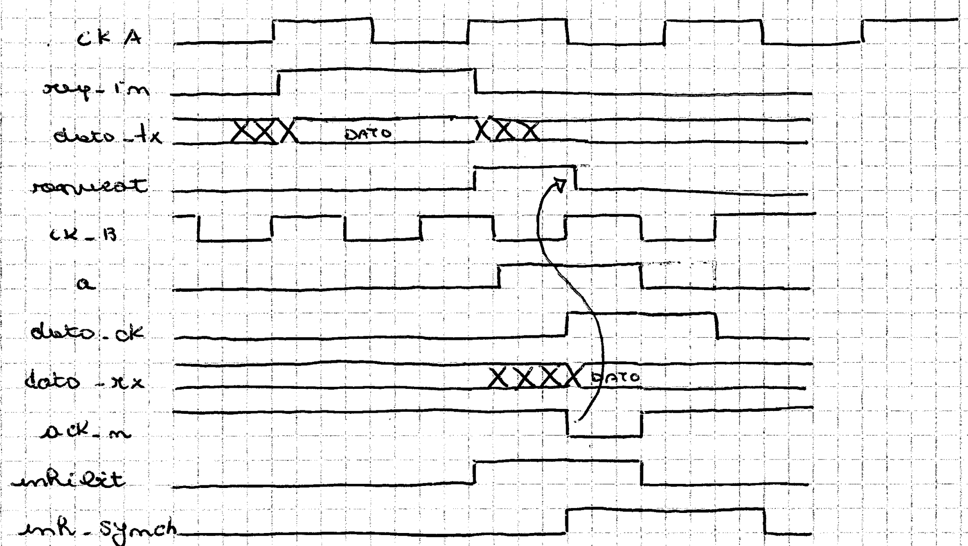
La soluzione è ricondurci ad un unico punto decisionale.

Infatti inh ↑ sincrono con A e inh ↓ sincrono con B!

Protocollo di Handshake protetto:



- Consente di perdere solo metà cicli di clock B.



Precedentemente è studiato il caso in cui Request non viola t_H o t_{su} di B.

Sono state viste più volte le regole del progetto sincrono al fine di ottenere massima banda di comunicazione.

inh ↑ sincrono con A e inh ↓ sincrono con $\overline{CLK\ B}$!
inh è un segnale asincrono rispetto ad A.

inh_synch sincrono con $\overline{CLK\ A}$.

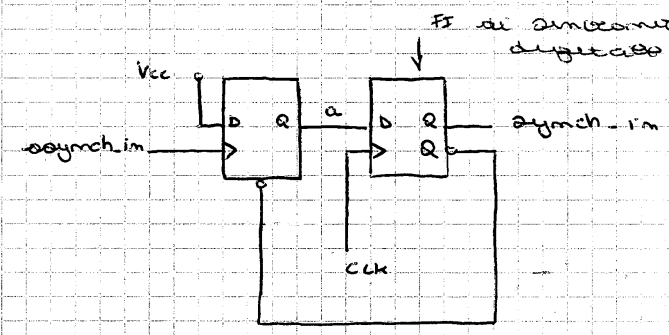
Supponiamo che il FF di protezione campioni male Request perché in violazione dei tempi di t_{su} o t_H.
Se $Q = 1$, abbiamo un intero mezzo ciclo di clock B perché B vada a regime ed invii dati esatti.
Se $Q = \emptyset$, la richiesta non è stata ancora rilevata.

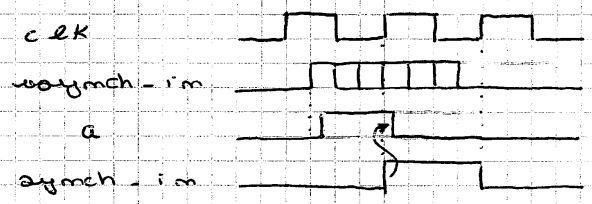
C'è comunque una probabilità di errore $\neq \emptyset$!
-> il FF di protezione deve concludere la metastabilità entro mezzo ciclo di CLK B.
Possibile soluzione: cascata di più FF di protezione
-> riducono il tempo in metastabilità grazie alla rigenerazione dei livelli logici.

06/12/2017

Le soluzioni precedenti sono valide a patto di mantenere per un certo intervallo di tempo.
-> il mio sistema è in grado di riconoscere un livello, non un fronte.

Come riesco a discriminare un fronte asincrono?



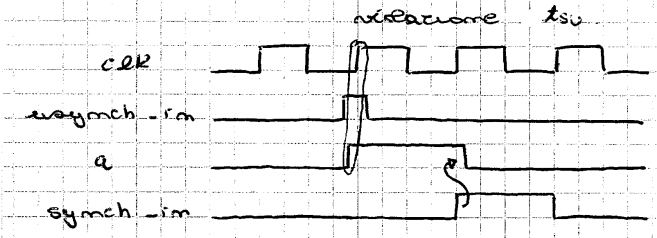


abbiamo realizzato un riconoscitore di impulso sincrono.

Siamo in grado di discriminare un fronte anziché un livello.

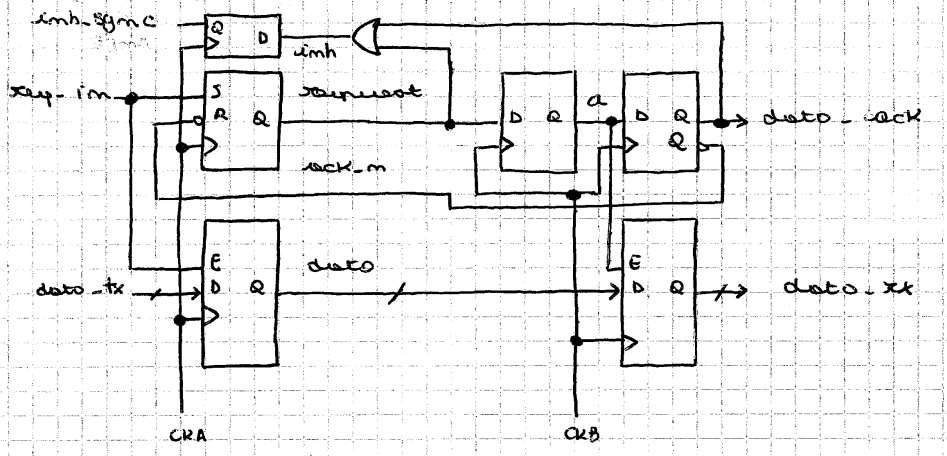
Se a noi vanno i tempi di t_{su} e t_H.
-> il campionamento è incerto.

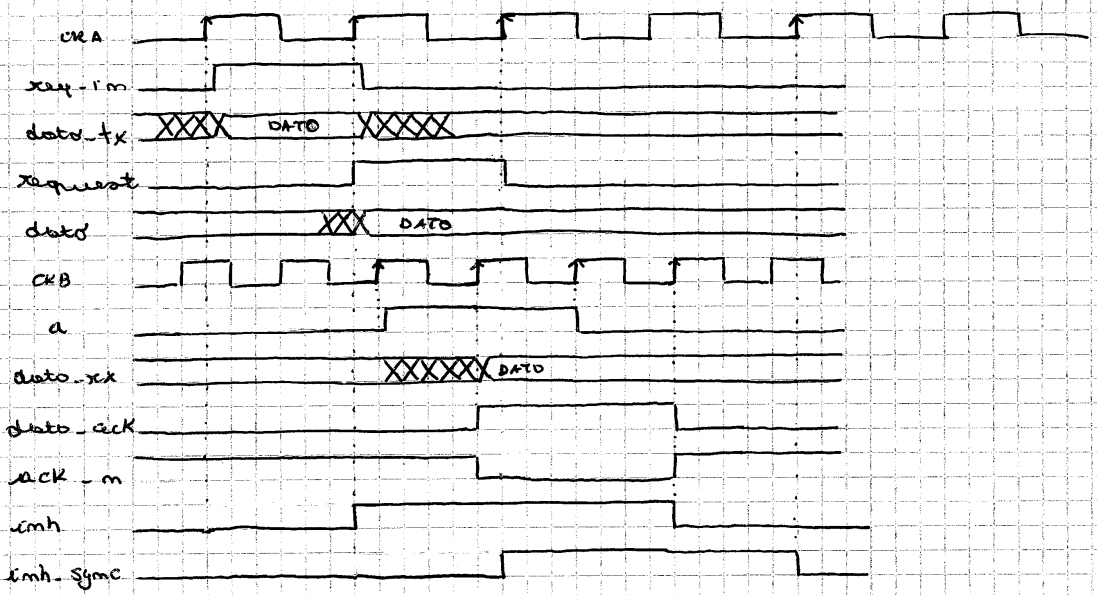
Ma ecco cosa accade in realtà:



Attengo comunque il risultato voluto.

nel mio sistema distribuito synch_in è non a, in quanto sincrono al sistema digitale.
Protocollo di Handshake che non vieda le regole del progetto sincrono statico:
Protocollo gentleman agreement.





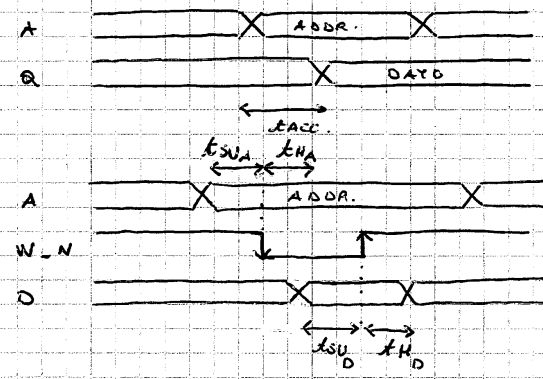
Si utilizzano FF di ricircola di tipo S̄R̄ anziché di tipo R.
ack_m determina il reset di request simultaneamente con CLK A.
Se Request viola t_H o t_{SU} del FF di protezione e Q = 1 ho un periodo di CLK B per stabilizzare i dati.

12/12/2017

Interfacciamento con memorie a semiconduttore.
memorie statiche e dinamiche.

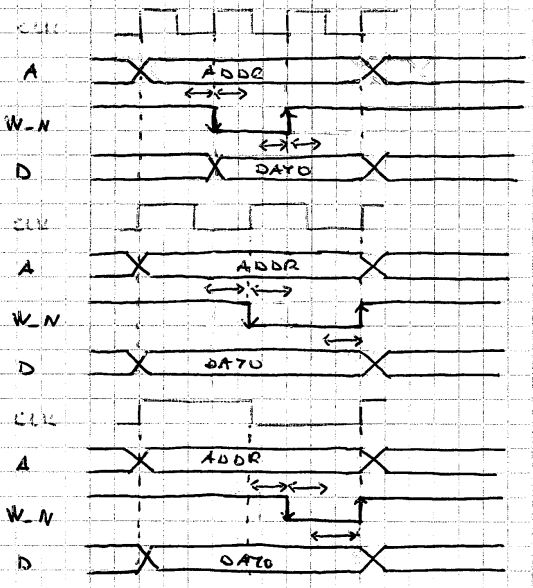
Le memorie statiche sono facilmente trovate all'interno dei chip in quanto realizzate con la stessa tecnologia dell'IC.

La prima cosa da fare per utilizzare una mem. statica è verificare la timing diagramm.
(SRAM)



Lettura

Scrittura: vanno rispettati t_H e t_{SU}



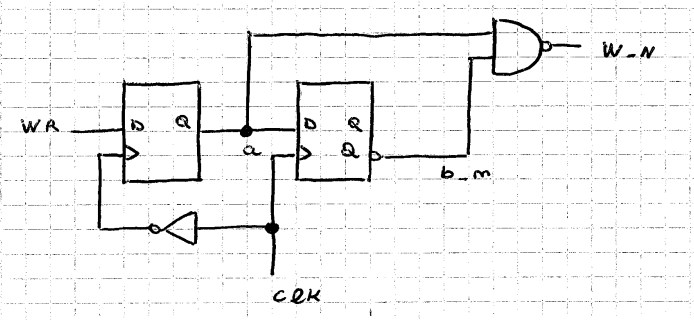
Temporizzazione a 3 cicli di clock ($t_H > \phi$)
-> nessun problema a rispettare t_{SU} e t_H.

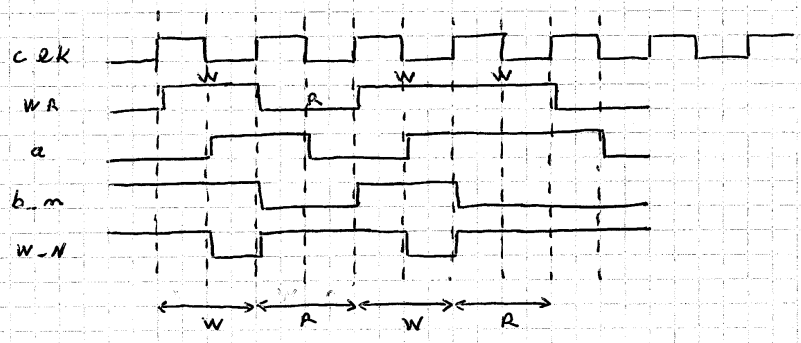
Temporizzazione a 2 cicli di clock ($t_H = \phi$)

Temporizzazione ad 1 ciclo di clock ($t_H = \phi$)
-> A e D sincroni con CLK ma W_N asincrono con $\overline{CLK}$.

Problema della scrittura ad 1 ciclo di clock: generazione fronte di discesa ↓ di W_N.
Esso deve essere ricavato all'interno del ciclo di clock -> necessaria una retroazione della

regola del progetto sincrono statico : DOUBLE EDGE CLOCKING. Si aggiunge un fronte di discesa di W_N al fronte di discesa di CLK.





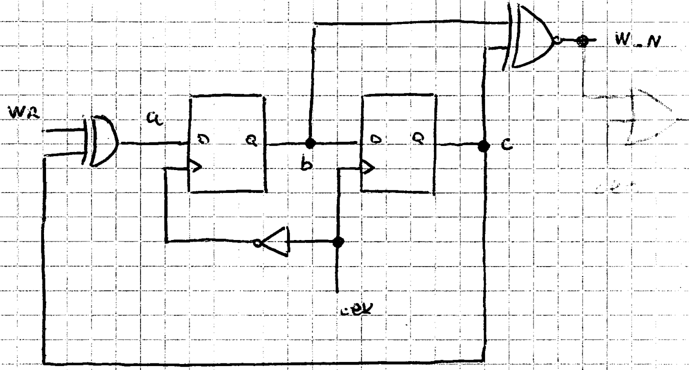
Sequenza di operazioni da eseguire : W, R, W, W.

La rete che genera WR ha solo mezzo ciclo di clock per mantenere a regime un cambiamento completato sul fronte negativo.

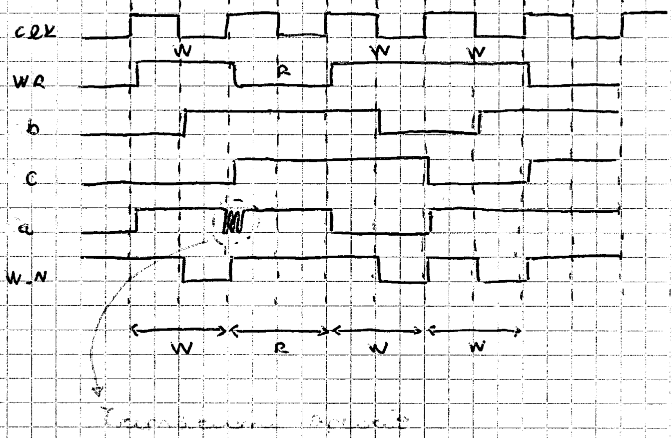
Sequenza di operazioni errata : W, R, W, R.

-> La rete non consente di fare due cicli di scrittura consecutivi.

Approccio leggi - scrivi supportato.



Si utilizza ancora la tecnica dual edge clocking

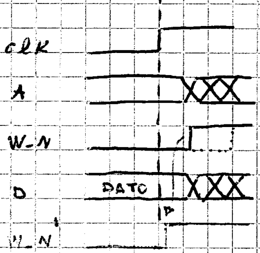


Sequenza di operazioni da eseguire: W, R, W, W.

Condizioni iniziali: $b = \phi$, $c = \phi$.

Sequenza di operazioni svolte: W, R, W, W.

nel caso $t_H = \phi$



← (ZOOM)

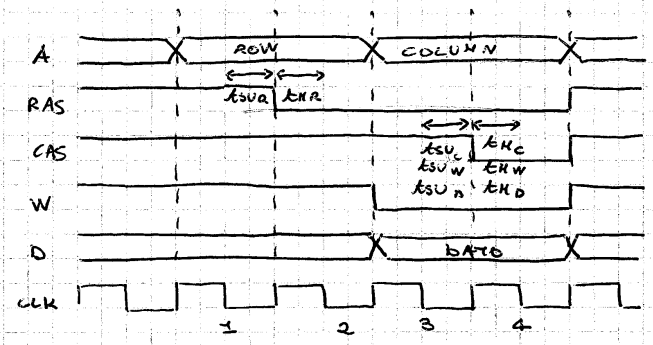
Il fronte ↑ di W_N deve avvenire prima del mutamento di A e D.

A e D hanno $t_H = \phi$ rispetto al fronte ↑ di W_N.

Nel caso ci sia una violazione timing è possibile anticipare il fronte ↑ di W_N inserendo clock gating.

→ W_N' anticipato rispetto a W_N.

Interfacciare una mem. dinamica asincrona.



Temporizzazione a 4 cicli di clock

Rispetta completamente il contratto comportamentale

di una DRAM di tipo asincrono.

Possibile temporizzazione a 2 cicli di clock con ridefinizione contestuale della logica del progetto sincrono statico.

Interfacciare una mem. dinamica sincrona SDRAM

-> Ci si riconduce ad interfacciare due sistemi mutuamente asincroni tra loro se clockati dallo stesso CLK.

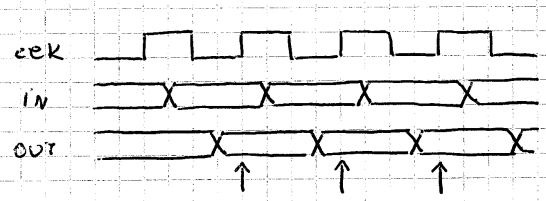
=> uso un'interfaccia di tipo sincrono.

Test del Progetto (esclusi)

metodi per riconoscere difetti di realizzazione da parte della fonderia

1) collaudo funzionale

2) collaudo strutturale



Si valutano le uscite

ATE : Automatic Test Equipment

# CICLO	IN	OUT - MIS	OUT - ASPETTATA
1	0x1x0x	0x0	0x0
2	0x1x0x	0x1	0 x 0 ← errore
3			
.			
.			

Qual'è la migliore sequenza di ingressi di test con cui collaudare il sistema ?
Im che condizioni operative svolgo questi test ?

1) Test Statici :
   - Guasti a livello fisico di clock
2) Test in velocità :
   - viene vista che effettivamente il circuito è testato ai frequenza $f_{CLK\ MAX}$
3) Test parametrici :
   - caratterizziamo delle caratteristiche elettriche del circuito

Scenario di test :
Si definisce il fault (condizione che definisce un guasto)
1) Stuck - @ - : meccanismo di guasto che blocca - no il valore di un nodo ad un valore predefinito.
   stuck - @ - 0 , stuck - @ - 1.
2) Single Fault : singolo guasto in tutto il circuito.

N - nodi -> numero possibile di guasti : 2N .
2N faults.

Metrica : Fault Coverage
Parametro che si assegna al vettore di test.

Identifica : # fault detected / 2N . → Fallimenti riconosciuti dalla sequenza di test.

Fault Simulation e Fault Injection:
si pone un single node ad un certo valore e si verifica che la sequenza di test rileva questo errore.

Sono necessarie 2N + 1 simulazioni:
- 2N per i nodi
- 1 per la fault free.

I tempi impiegati per la simulazione sono inaccettabili.

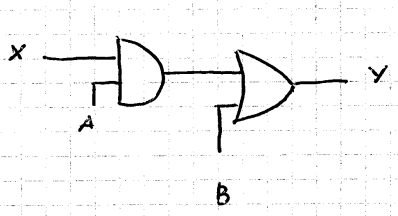
Invece di fare un fault simulation sequenziale faccio un fault simulation statistica.

Un nodo, per essere verificabile, deve essere OSSERVABILE (è uno stato influenza il valore dell'uscita) e CONTROLLABILE (stato influenzato dal valore dell'ingresso).

un nodo è testabile se è osservabile e controllabile.

Rendere un nodo osservabile:

es.

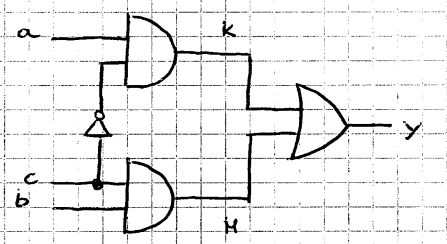


Perché x sia osservabile è necessario che:
A=1
B=Ø ← condizioni di sensibilizzazione del cammino X -> Y

Sono necessari ingressi che attivino le condizioni di eccitazione.

A tal punto è necessario controllare il nodo.

Esempio:



Individuare stack_@_$\not$ in k.

1) OSSERVARE K

- k = $\not$, b = $\not$ c = x
- k = $\not$, b = x c = $\not$

2) CONTROLLARE K

k = 1

- a = 1 , c = $\not$

Condizioni ingressi:

a = 1	1	1	verifichiamo entrambi stack_@_$\not$ in k.
b = x	1	0	
c = $\not$	0	0	

ATPG: Automatic Test Pattern Generator

→ Sistemi progettuali automatici che creano i migliori vettori di test.

È necessario utilizzare tecniche di progetto orientati alla DFT (Design for Testability).

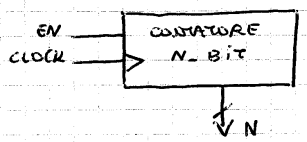
13/12/2017

Design for Testability

Hanno lo scopo di rendere tutti i nodi controllabili in maniera semplice.

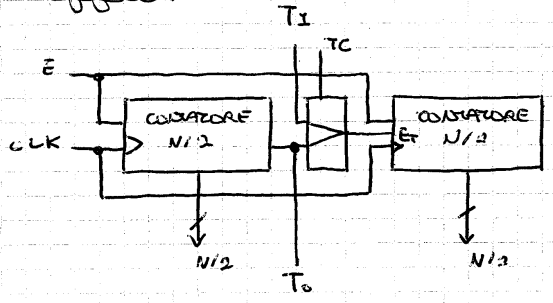
- tecniche ad hoc •;
- tecnica strutturata ;

es. Contatore N-bit



È una struttura difficilmente controllabile.
-> richiede molti cicli di test : 2^N.

• una soluzione è interruzione della catena di conteggio.

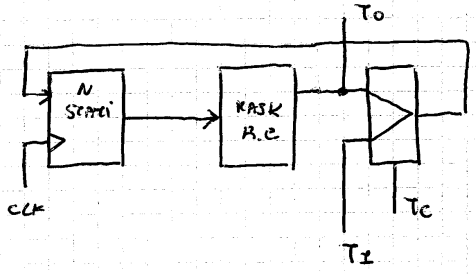


TI : test input
TC : test control
TO : test output

Con questa soluzione per controllare l'ultimo bit del contatore sono necessari $2^{\frac{N}{2}}$ cicli

C'è una modalità di test e una modalità di funzionamento a seconda del valore di Tc.

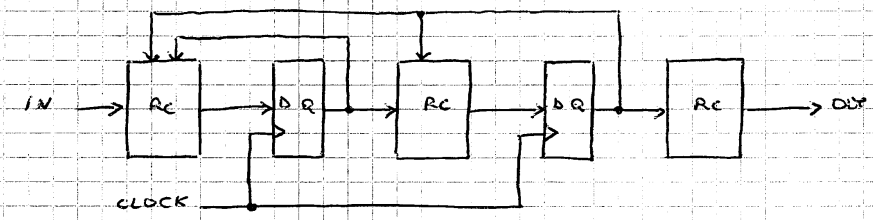
es. Macchina in cascata



• DEGATING ANELLO DI REAZIONE

Consente di portare facilmente la macchina in un certo stato.

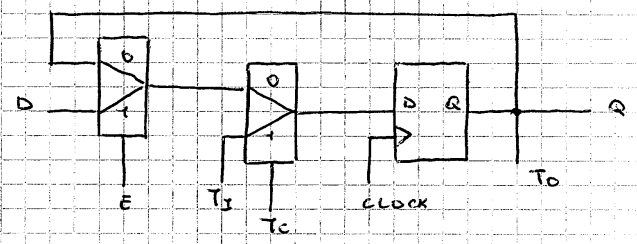
Tecniche strutturate



Le celle sono caricate i registri con un pattern di segnali di controllo ed sono accedo alle uscite delle celle RC.

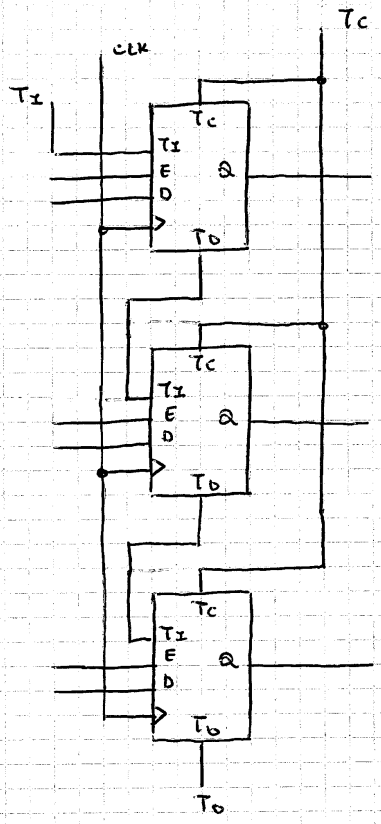
Tecnica Scan - Path

Si usa uno Scan FLIP FLOP :
SCAN FF



Si sostituiscono tutti i FF con gli SCAN FF, che in modalità di funzionamento non alterano la funzionalità del sistema.

Analizziamo uno dei registri del sistema



In condizioni $T_C = 1$ questa struttura è un grande registro a scorrimento. Dm N cicli di clock in modalità di test riesco a caricare i registri con valore desiderato.

Facendo funzionare gli SCAN FF per 1 ciclo in modalità di funzionamento i registri memorizzano tutte le uscite della rete combinatoria.

Applicando N colpi di clock in modalità test riesco a far scorrere il risultato verso l'uscita T_o del sistema.

Ho applicato un test pattern ai registri ed Ho misurato il valore delle uscite

delle RC.

Per eseguire un test pattern sono necessari $2N+1$ colpi di clock.

→ in realtà sono sufficienti $N+1$ colpi di clock: durante lo spostamento passo di nuovo caricare i registri.

I PIN accessibili dall'esterno per poter pilotare questo registro a scorrimento si chiamano JTAG.