Reti logiche

- combinatorie: output dipende da input
- sequenziali: output dipende da input presenti e passati

sintesi

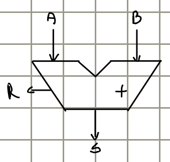
Funzione → espressioni/Formule → schema logico

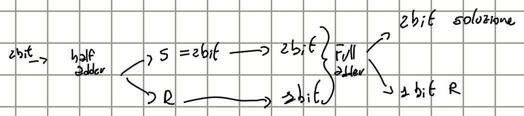
analisi

schema logico → espressioni / Formule → Funzione

half adder

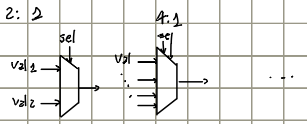
→ somma i bit di A e B con resto



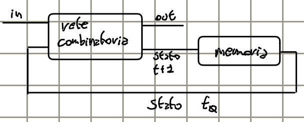


mux

→ n: 1 n input, 1 solo output, e bit di selezione



rete sequenziale



latch-SR

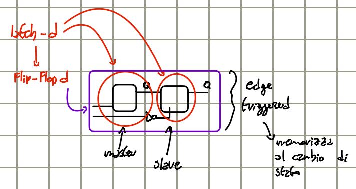
set
reset

SR	
00	stato in memo Q
01	0 reset
10	1 set
11	--

latch-d

↓

Flip-Flop d



edge triggered → memorizza al cambio di stato

Flip-Flop SR

SR	Q_{n+1}
00	Q
01	0
10	1
11	x

FF JK

JK	Q_{n+1}
00	Q_n
01	0
10	1
11	$\overline{Q_n}$

FF T

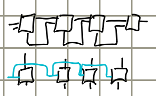
→ Toggle

↓

fa il set 1 al cambio di stato del bit

registri

- sono composti da n Flip Flop
  - a scorrimento
  - in parallelo



FSM

- sincrone: clock
- asincrone: no-clock

diagramma stati

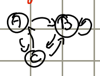


tabella stati

	x	
S	0	1
A	A,1	B,0
B	B,0	A,1
C	A,0	B,0

codifica stati

A=00
B=01
C=10

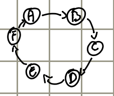
tabella stati + codifica stati → tabella transizioni con codifica stati

reti costo minimo

↓

uso mappe k

contatore

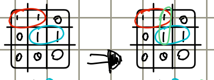


alee

↓

output errati dovuti ai diversi tempi di esecuzione dei circuiti

- come evitare



A + B
mappa k con alee

A + B + C
mappa k senza alee

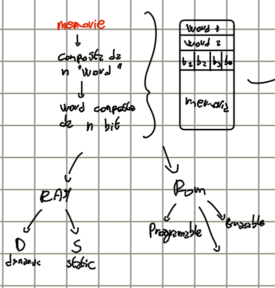
memorie

↓

composta da n "word"

↓

word composta da n bit



decoder

il decoder manda a 1 una word e le altre a 0, permettendo così di leggerle

↓

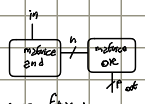
fungono da selettori multiplexer della memoria

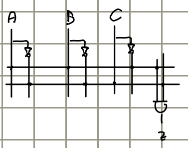
PLA

Programmable Logic array

↓

usata x esprimere funzioni





VHDL

H S I C (hardware description language)

Progetta sistemi digitali definendo comportamento logico e digitale — non é case sensitive

↳ consente di: modellare processi in parallelo → operazioni in simultaneo

Testbench → modulo di test per verifica e validazione della funzionalità di un progetto

Entity → descrive interfaccia di un circuito specificando in/out, nome del componente → un entity può avere più architecture

```
entity nome is
   Port(
      a: in std_logic; -- input
      b: out std_out; -- output
   )
end nome;
```

architecture → descrive comportamento e struttura del circuito definito nell' entity

```
architecture comportamento of nome is
begin
   b<= not a; -- comport...
end comportamento;
```

Port → definisco le connessioni: in / out / inout

STd_logic → segnale che può essere: 0 / 1 / z / x "indefinito" / Uninitialized

↓

STd_logic_vector

y <= "1011";

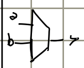
statement → sono le istruzioni, terminano con ";"

↳ assegnazione a <= b → ad "a" assegno il val di "b"

librerie → Funzioni / definizioni } si includono: library IEEE; use IEEE. quello che ti serve;

mux 2:1

```
entity Mux is
   Port(
      a,b : in std_logic; -- input
      y : out std_out; -- output
   )
end Mux ;
```



sensitive list → lista di segnali che attivano un processo al loro cambiamento

Port Map → utilizzato per collegare un componente ai segnali nel design → collega i port di un entità a segnali o variabili all' interno dell' architettura di un altro componente, permettendo di istanziare entità e di mapparne i segnali

```
and_gate port map(
      a => input 1,
      b => input 2,
      y => output
);
```

type → comando per definire tipi personalizzati

```
type stato is (s0,s1,s2,s3);
signal stato_corrente, stato_futuro: stato;
```

Signal dichiara segnali di un particolare tipo

Registro a scorrimento

↳ reg(7 downto 0) <= S_in & reg(7 downto 1);

nuovo bit che deve entrare ↓ S_in

assegnazione Hex

var <= x"56"

↑ trasforma il valore 56 da Hex a bin

Progettazione gerarchica

↓

divide il progetto in sotto moduli più piccoli

base

```
entity half_adder is
   Port(
      a,b : in std_logic;
      sum, carry: out std_logic;
   );
end half_adder;
```

composizione

U1: half_adder Port map(a => input1, b => input2, sum => output1, carry => carry1);

Provo a rispondere a domande prof

1) VHSic hardware description language
2) un linguaggio per descrivere a livello funzionale e logico l' hardware e simularlo
3) permette la descrizione di processi in parallelo
4) verificare e convalidare le funzionalità del programma
5) l' entity è l' istruzione che permette di instanziare un entità con variabili al suo interno
6) fornisce la logica all' entity
7) definisce il comportamento delle var dell' entity in, out, inout
8) 0,1, z, x, U
   x ↑ undefined
9) '0'
10) un entity può avere più architecture
11) un entity ha diverse o una sola port
12) il ; si mette per concludere una istruzione
13) library libreria; use libreria.~~~;
14) mux 2:1
15) 2^n, nella prima c'è 1 spazio in più anche se non cambia nulla
16) inout
17) in <= out
18) ↗
19)
20) out <= not (in)
21) y <= "1011"
22) Per partire determinati processi al mutamento di una variabile nella SL, riducendo il carico computazionale
23) ? la port map
24) definisci tipi personalizzati e con signal li assegni
25) ? con reset sincro attivo alto
26) registro a scorrimento
27) mux 4:1
28) x <= x"56" Conversione di 56 da Hex a bin
29) -----
30) riutilizzo, mantenibilità, cooperabilità e chiarezza del codice