Modellazione strutturale

▼ Creatore originale: @Gianbattista Busonera

Definizione di un modulo
    Esempio - Interfaccia del modulo ALU
    Esempio - Interfaccia di un processore

La modellazione strutturale descrive la struttura gerarchica del circuito tramite istanze di moduli (che possono a loro volta essere RTL, gate-level, etc.), è simile a disegnare uno schematico, ma con una descrizione testuale.

L'elemento base è `module .... endmodule` per ogni blocco, le cui istanze si collegano con net (tipicamente `wire`).

```
module <nome> ([lista delle porte]);
   // contenuti del modulo
endmodule

// si noti come un modulo può omettere la
// lista delle porte
module <nome>;

endmodule
```

Tutte le connessioni sono simultanee, e quindi l'ordine delle istanze non conta.

Il loro uso tipico è:

- integrare blocchi IP (Intellectual Property block), ovvero blocchi di logica già progettati, testati ed ottimizzati, che è possibile comprare, licenziare o riutilizzare all'interno di un progetto invece che implementarlo da zero;
- costruire sistemi gerarchici (per esempio, CPU + periferiche).

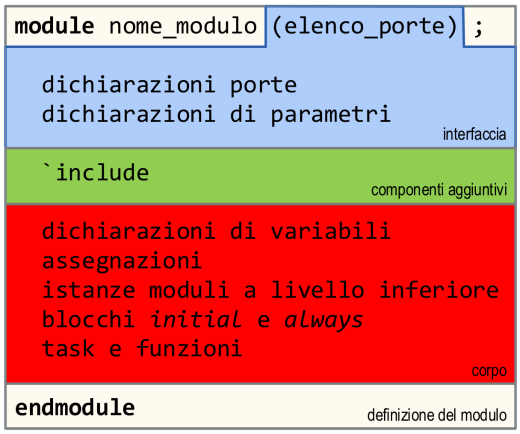
Nella pratica moderna, il 90% del codice è scritto in RTL o dataflow, per poi lasciare al sintetizzatore la generazione gate-level finale. Si usa, invece, la modellazione strutturale per mettere insieme i vari moduli.

Definizione di un modulo

Un modulo è un componente riutilizzabile. In Verilog, ogni blocco di circuito è racchiuso tra le parole chiave `module` ed `endmodule` .

All'interno del modulo si dichiarano:

- elenco degli ingressi e delle uscite (le uscite vanno inserite prima per convenzione);
- le caratteristiche delle porte (ingressi e uscite, eventualmente signed) e il loro parallelismo;
- eventuale dichiarazione di parametri (costanti di cui possiamo modificare il valore), che ci consentono di rendere parametrico il nostro circuito;
- tramite la parola chiave `` `include `` possiamo inserire nel nostro modulo altri componenti definiti in precedenza;
- segue la parte più corposa del nostro modulo, che analizzeremo man mano nella trattazione.



Visualizzazione della struttura interna di un modulo

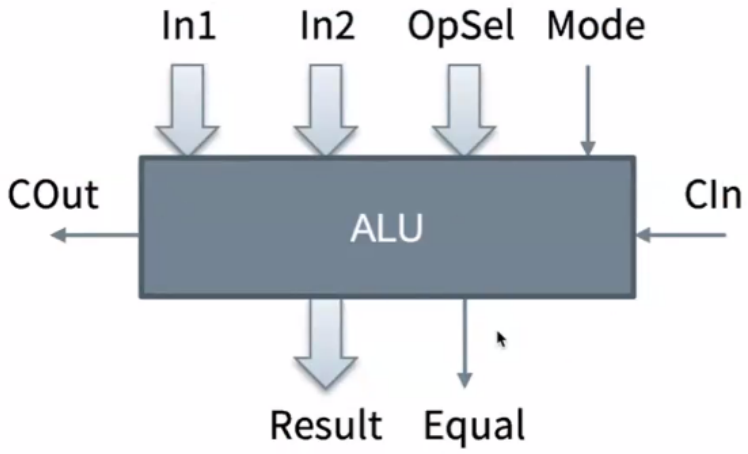
```
module  nome_modulo
        #(parametri opzionali)   // facoltativo: parametri generici
        (elenco_porte);          // porte fra parentesi tonde
    /* dichiarazioni */
    // 1. dichiarazione delle porte:
```

```
    //    input, output (signed) , inout(+ eventuale larghezza bus)
    // 2. dichiarazione di segnali locali (wire, reg, logic, ecc.)
    // 3. descrizione della logica con:
    //    - primitive di porta          (gate-level)
    //    - assign continui             (data-flow / RTL)
    //    - blocchi always/initial      (behavioral / RTL)
endmodule
```

▼ Esempio - Interfaccia del modulo ALU

Si progetti, tramite Verilog, l'interfaccia del modulo ALU in figura.

Ipotizziamo che questa ALU possa fare solo operazioni di somma e sottrazione e che gli operandi `In1` e `In2` siano su 3 bit.



Visualizzazione di un modulo ALU

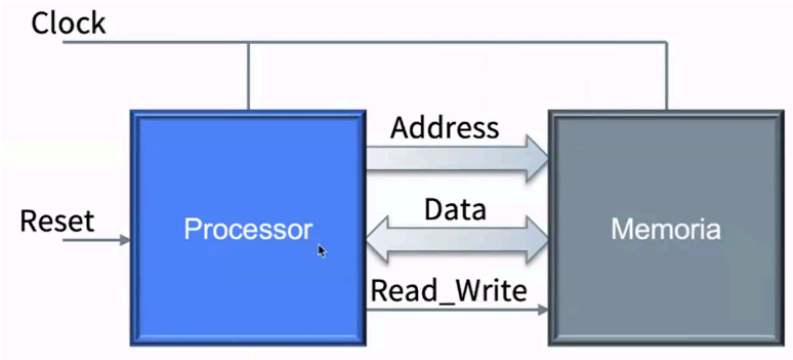
```
module ALU (Result, COut, Equal,        // prima le uscite, per convenzione
                    In1, In2, OpSel, CIn, Mode); // seguite dagli ingressi.
                                                 // si dichiarano esplicitament
                                                 // tipo delle "porte" (ingressi/
                                                 // e il parallelismo

output [4:0] Result; // porta d'uscita Result su 4 bit
                                 // in caso di somme di numeri a 3 bit potremmo
                                 // ottenere un numero su 4 bit

output COut;      // carry out
output Equal;     // se Equal = 1 ⇒ In1 = In2
```

```
input [2:0] In1;   // primo operando
input [2:0] In2;   // secondo operando
input [2:0] OpSel; // ipotizziamo ci siano 8 operazioni possibili
input CIn;
input Mode;        // modalità aritmetica (1) o logica (se 0)
// FINE INTERFACCIA
...
endmodule
```

▼ Esempio - Interfaccia di un processore

Si progetti, tramite Verilog, l'interfaccia del modulo Processore in figura.

Tale processore ha la capacità di leggere o scrivere in memoria, ed è comandato da un clock.



Visualizzazione di un modulo Processore

```
module Processor (Read_Write, Data, Clock, Reset, Address);

output Read_Write;
output [19:0] Address;
inout [15:0] Data; // è sia un input che un output!
                   // Il processore può leggere o scrivere dati in/da memor
input Clock;
input Reset;
// FINE INTERFACCIA
...
endmodule
```

⚠️ Normalmente, le porte di tipo `inout` dovrebbero essere pilotate con porte tri-state, in modo da evitare la creazione di conflitti.