2.6 Riflessione conclusiva su descrizione e sintesi delle reti logiche

Abbiamo visto vari tipi di reti logiche: quelle combinatorie, sia semplici (pochi ingressi ed uscite) sia complesse (e.g., quelle per l'aritmetica, caratterizzate da molti ingressi ed uscite); quelle sequenziali asincrone; quelle sequenziali sincronizzate, sia semplici (cioè con pochi ingressi, stati interni ed uscite) che complesse (le ultime che abbiamo visto). Siamo al punto giusto per riprendere in mano il progetto di una rete logica con maggior cognizione di causa.

Una rete logica deve essere **prima descritta**, e **poi sintetizzata**.

La **descrizione** altro non è che un **modo formale** di fornire le **specifiche del comportamento** della rete medesima. Infatti:

- Nel caso di una **rete combinatoria**, è un'associazione tra stati di ingresso e stati di uscita, per esempio scritta sotto forma di tabella di verità.
- Nel caso di una **rete sequenziale** (asincrona o sincronizzata), cioè di una rete con memoria, è un **diagramma a stati**, che può essere rappresentato:
    a) tramite tabella o grafo se la rete è abbastanza semplice, oppure
    b) tramite un formalismo più complesso, in cui ad ogni stato interno vengono associate delle azioni che la rete esegue (nel nostro caso, assegnamenti a registri).

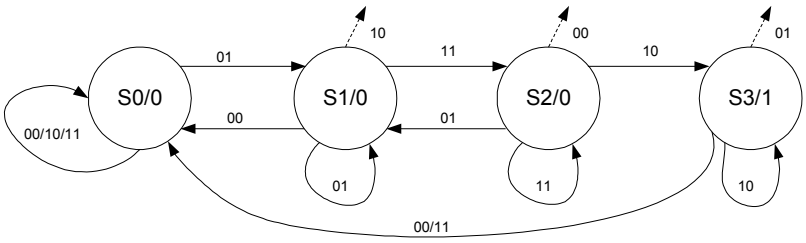
Per descrivere una rete esistono **formalismi differenti**, che si adattano meglio o peggio ad un particolare tipo di rete**.** Anche per uno stesso tipo di rete si possono usare più formalismi diversi. Ad esempio, per una rete combinatoria possiamo usare indifferentemente una **tabella di verità** o **poche righe di Verilog**.

x2	**x1**	**x0**	**z**
0	0	0	0
0	0	1	1
0	1	0	0
…	…	…	…

```
module Rete(x2, x1, x0, z);
  input x2, x1, x0;
  output z;
  assign z=({x2,x1,x0}== 'B000)?    'B0:
           ({x2,x1,x0}=='B001)?     'B1:
           ({x2,x1,x0}=='B010)?     'B0:
            …
endmodule
```

È chiaro che la tabella di verità e la descrizione in Verilog scritta accanto hanno la medesima semantica. È altresì chiaro che entrambe dicono **cosa fa** la rete, **ma non come è realizzata,** cioè quali sono le porte logiche che la compongono.

Allo stesso modo, la descrizione di una RSA si può dare sotto forma di grafo di flusso, di tabella di flusso, o di linguaggio Verilog. Quest'ultimo non l'abbiamo mai usato, ma è chiaro che avremmo potuto farlo.



x_1x_0	00	01	11	10	z
S0	(S0)	S1	(S0)	(S0)	0
S1	S0	(S1)	S2	--	0
S2	--	S1	(S2)	S3	0
S3	S0	--	S0	(S3)	1

È indispensabile avere a disposizione un modo formale per descrivere una rete, perché una descrizione formale può essere **verificata**:

- verificare la descrizione di una RC significa controllare che gli stati di uscita siano quelli desiderati per ogni possibile stato di ingresso. È una verifica **statica**, che si fa per ispezione diretta.
- Verificare la descrizione di una RS (asincrona o sincronizzata) significa **simulare l'evoluzione della rete** a partire da una condizione iniziale di reset, e controllare che questa sia coerente con le specifiche (normalmente date a parole). Questa è una verifica **dinamica** (richiede un'evoluzione temporale della rete), concettualmente simile al processo di testing di un software. Così come è difficile, se non impossibile, testare il software in maniera esaustiva (i.e., per tutti i possibili input), è difficile verificare in maniera esaustiva il comportamento di reti sequenziali che non siano estremamente semplici. Nondimeno, è **sbagliato** non verificare la descrizione di una RS, tanto quanto è sbagliato non testare del software. Un **diagramma di temporizzazione** che mostra la simulazione di una RSS complessa con un certo input, come quelli che abbiamo fatto svariate volte finora, è un modo per verificare la descrizione di quella rete.

Ciò che rende un **formalismo di descrizione** preferibile rispetto ad un altro è **quanto è comodo da usare**. Per una RC semplice una tabella di verità o una mappa di Karnaugh sono (leggermente) più comodi di una descrizione in Verilog. La verifica statica di una tabella di verità è (leggermente) più agevole rispetto a quella di una descrizione in Verilog.

Allo stesso modo, per una RSA (semplice) una tabella di flusso è più comoda da scrivere di una descrizione in Verilog, e soprattutto è più facile – per un utente umano – simulare l'evoluzione della rete sulla tabella di flusso che sulla descrizione in Verilog. Se invece volessi avvalermi di un simulatore Verilog per simulare il comportamento di una RSA, mi converrebbe ovviamente scriverne la descrizione in Verilog.

Viceversa, per una RSS complessa, una descrizione in Verilog risulta ben leggibile, e può essere agevolmente fornita ad un simulatore Verilog per verificare il comportamento della rete.

Una descrizione non può fornire informazioni su **come è realizzata la rete**, cioè quali sono i suoi componenti elementari e come sono interconnessi. Per arrivare a questo livello è necessario procedere alla **sintesi**, che è **sempre** il punto di arrivo degli esercizi che svolgiamo. Fermarsi alla descrizione, infatti, comunica l'impressione di **non essere in grado di realizzare** ciò che si è pensato, impressione che un ingegnere non dovrebbe mai dare.

Esistono **due approcci** per la sintesi, e li abbiamo usati entrambi: quello **euristico**, che consiste nel "fare le cose ad occhio", e quello **formale**, che consiste nel seguire un procedimento algoritmico. L'approccio euristico viene usato tipicamente nel caso di **reti combinatorie per l'aritmetica**. La sintesi richiesta negli esercizi di aritmetica consiste, di fatto, nel prendere dei blocchi "atomici", e.g., sommatori, moltiplicatori, etc. – la cui struttura interna è data per assodata – ed assemblarli in modo che soddisfino delle specifiche date a parole (il testo dell'esercizio). Approcci euristici per la risoluzione di problemi sono tipici del know-how di un ingegnere (il processo di scrittura del software a partire dalle specifiche è, infatti, un procedimento euristico), e richiedono l'esperienza che si acquisisce solo con la pratica.
Abbiamo anche usato **approcci formali** per la sintesi delle reti logiche. Ad esempio:

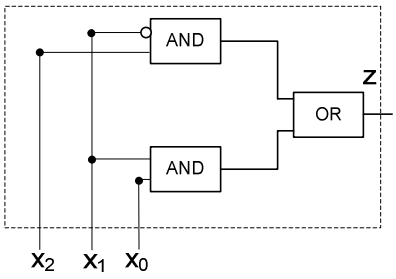
- La sintesi a costo minimo in forma SP (PS, NAND, NOR) di una RC;
- la sintesi di una RSA secondo il modello con elementi neutri di ritardo (o con latch SR) come elementi di marcatura;
- la sintesi di una RSS di Moore, Mealy, Mealy Ritardato, secondo il modello con registri FF-D o FF-JK;
- la sintesi di una RSS complessa secondo il modello con scomposizione in Parte Operativa e Parte Controllo.

In tutti questi casi, il processo seguito è di tipo algoritmico. Si parte dalla **descrizione** della rete, si adotta un **modello di sintesi**, e si procede secondo i passi dell'algoritmo. La scelta del modello di sintesi determina l'algoritmo. Ad esempio, l'algoritmo di sintesi di una RSA a partire dalla descrizione è diverso a seconda che scelga di usare il modello con elementi neutri di ritardo o quello con latch SR come elementi di marcatura. Allo stesso modo, la sintesi di una RSS di Moore è diversa se scelgo di usare il modello con registro di stato FF-D o quello con registro di stato FF-JK. L'algoritmo di sintesi di una RC è diverso a seconda che scelga un modello di sintesi SP o PS.

Sia come sia, alla fine una **sintesi** deve essere comunicata a qualcuno (e.g., il tecnico che realizzerà l'hardware) in forma intelligibile, e quindi deve essere **scritta secondo un qualche formalismo**

pure lei. Come per le descrizioni, abbiamo usato diversi formalismi, scegliendo di volta in volta quello **più comodo perché più facile da leggere**.

Ad esempio, per le RC semplici abbiamo usato indifferentemente **diagrammi, espressioni di algebra di Boole, o il linguaggio Verilog**.



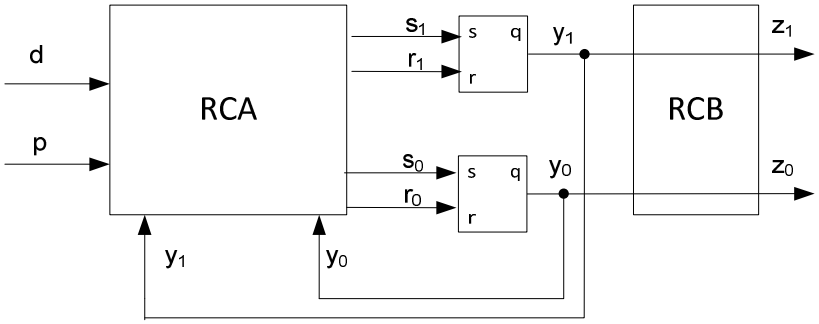

```
module Rete(x2, x1, x0, z);
  input x2, x1, x0;
  output z;
  assign z= (x1 & x0) |
            (x2 & ~x1);
endmodule
```

$$z = (x_1 \cdot x_0) + (x_2 \cdot \overline{x_1})$$

Tutti e tre sono equivalenti, ma noi preferiamo usare le espressioni algebriche perché sono più compatte. Si noti che il linguaggio **Verilog** può essere usato sia per rappresentare la **descrizione che la sintesi**. Ciò è spesso fonte di confusione, perché si confonde il **formalismo** (cioè il linguaggio Verilog) con il **contenuto** (cioè la descrizione o la sintesi, a seconda dei casi). È però chiaro che il pezzo di Verilog scritto sopra è una **sintesi**, in quanto è mappabile direttamente su porte logiche interconnesse. Non è una descrizione perché non dice come stati di uscita corrispondono a stati di ingresso.

Per la sintesi di RC complesse (quali quelle per l'aritmetica) usiamo normalmente **diagrammi**. Più raramente, scriviamo sintesi in Verilog (sono più difficili da leggere).

Per le RSA e le RSS di Moore, Mealy, e Mealy Ritardato usiamo normalmente un **diagramma** che specifica quale **modello di sintesi** abbiamo scelto, ed **espressioni algebriche** che descrivono le relazioni ingresso-uscita delle reti combinatorie facenti parte del modello.



$$s_1 = p \cdot d \cdot \overline{y_0} = \overline{\overline{p \cdot d \cdot \overline{y_0}}} = \overline{\overline{p} + \overline{d} + y_0}\,,\ r_1 = \overline{p}$$

$$s_0 = p \cdot \overline{d} \cdot \overline{y_1} = \overline{\overline{p \cdot \overline{d} \cdot \overline{y_1}}} = \overline{\overline{p} + d + y_1}\,,\ r_0 = \overline{p}$$

$$z_1 = y_1\,,\ z_0 = y_0$$

Per le RSS complesse, è di gran lunga più conveniente usare il **Verilog** per scrivere la sintesi, sempre stando attenti a non confondere il formalismo con il contenuto, visto che usiamo il Verilog **anche** per scrivere la descrizione. Scrivere la sintesi sotto forma di diagramma in questo caso richiederebbe fogli enormi e grossi intrighi di fili, e ne risulterebbe qualcosa di difficile da leggere. **Resta inteso**, in ogni caso, che ciò che si scrive quando si fa una sintesi in Verilog è **in corrispondenza biunivoca con un diagramma**, in cui:

- la parte operativa è composta da registri multifunzionali, che hanno le variabili di comando b_j come ingressi di comando dei propri multiplexer;
- ci sono reti combinatorie che generano le variabili di condizionamento;
- la parte di controllo consiste in un registro di stato ed una ROM (e poco altro), ed ha in ingresso le variabili di condizionamento e in uscita quelle di comando.

Non aver capito questo significa non aver capito cosa si sta facendo, ed è **grave**. Per rendere chiaro che si è capito, negli esercizi di esame è **sempre** richiesto di disegnare almeno i diagrammi delle reti combinatorie di condizionamento e di specificare il contenuto della ROM sotto forma di tabella di verità. Talvolta, può essere richiesto di disegnare anche alcune porzioni della parte operativa.

Fare una **sintesi** di una rete sequenziale senza averne fatto la descrizione (errore che capita di vedere talvolta durante la correzione dei compiti) è cosa completamente **assurda**: **è praticamente impossibile inferire** il comportamento della rete a partire da una sintesi, e quindi non si riesce a verificare la correttezza della medesima rispetto a delle specifiche date.
Come nota a margine, si osserva che per una **rete combinatoria** è invece relativamente semplice risalire alla descrizione (e.g., alla tabella di verità) a partire dalla sintesi – il che non è comunque un buon motivo per saltare la descrizione. Ciò è dovuto al fatto che le reti combinatorie non hanno memoria.

Una sintesi **deve quindi essere coerente con la descrizione che l'ha prodotta,** in modo tale che la correttezza del comportamento della rete (che si verifica sulla descrizione) sia mantenuta nella implementazione della medesima. I procedimenti formali per sintetizzare le reti (quelli elencati sopra) partono infatti da una descrizione e la realizzano secondo un modello. Tali procedimenti **garantiscono** che la rete così sintetizzata si comporti nel modo specificato dalla sua descrizione.
Ad esempio, una RSA sintetizzata a partire dalla tabella di flusso secondo il modello con elementi neutri di ritardo, con ritardi dimensionati opportunamente, stati interni codificati opportunamente, e rete RCA priva di alee, si comporta come specificato nella tabella di flusso, se pilotata correttamen-

te. Analogamente, una RSS complessa sintetizzata secondo il modello PO/PC a partire dalla descrizione, si comporterà come la descrizione, purché il clock sia dimensionato in modo corretto e gli input non vengano modificati a cavallo dei fronti di salita del clock.
Alcune ottimizzazioni in fase di sintesi sono talvolta possibili. Si deve tener presente, però, che le sintesi non si giudicano dal livello di ottimizzazione: se fosse necessario ottenere una sintesi "di costo minimo", qualunque cosa questo voglia dire, lo si farebbe fare ad un programma.

Ricapitolando: il progetto di una rete logica (qualunque) si affronta nel seguente modo:

1. **descrizione**, per stabilire in modo formale (e quindi verificabile) qual è il comportamento della rete;
2. **sintesi**, a partire dalla descrizione e seguendo un apposito modello, per realizzare una rete che si comporta come specificato nella descrizione.

Per rappresentare i risultati di entrambi i passi si usano dei **formalismi**, quelli che meglio si adattano al tipo di rete. Il fatto che lo stesso formalismo (in particolare, il Verilog) si possa usare per entrambi i passi non può essere fonte di confusione, se si è capito cosa si sta facendo. La scelta del formalismo da usare nella descrizione e nella sintesi risponde a criteri di facilità di utilizzo ed economicità di spazio. In particolare:

- per le RC (semplici), una descrizione come tabella di verità è più leggibile. Una descrizione come mappa di Karnaugh è equivalente, e facilita il procedimento di sintesi secondo uno qualunque dei modelli noti. La sintesi si dà sotto forma di espressioni algebriche.
- per le RSA e le RSS di Moore, Mealy e Mealy ritardato con pochi ingressi e pochi stati interni, una descrizione come tabella di flusso è facile da verificare, ed inoltre facilita il procedimento di sintesi secondo uno dei modelli noti. La sintesi si dà sotto forma di indicazione del modello da utilizzare, più le espressioni algebriche delle uscite delle reti combinatorie facenti parte del modello.
- Per le RSS complesse, una descrizione in Verilog è facile da verificare, ed inoltre facilita il procedimento di sintesi secondo il modello con scomposizione in PO/PC. La sintesi si dà parimenti in Verilog, con alcuni diagrammi e tabelle di verità a completamento.