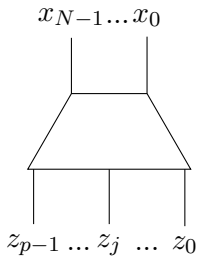
1 Lezione del 09-10-24

1.1 Decoder

Un decoder è una rete con N ingressi e p uscite con $p = 2^N$. Si indica come:



La sua legge di corrispondenza stabilisce che ogni uscita riconosce uno ed un solo stato di ingresso, in particolare l'uscita j-esima (z_j) riconosce lo stato di ingresso i cui bit sono la codifica di j in base 2, cioè:

$$(x_{n-1}, ..., x_0)_2 = j$$

Ad esempio, un decoder da 2 a 4 ha tabella di verità:

x_1	x_0	z_0	z_1	z_2	z_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

che equivale alla codifica *one-hot* del binario in ingresso (cioè ogni numero codificato da n bit viene mandato al j-esimo di p output che corrispondono uno ad uno ai numeri rappresentabili).

Vediamo di passare da questa descrizione ad una sintesi della rete. Abbiamo che:

$$\begin{cases} z_3 = x_1 \cdot x_0 \\ z_2 = x_1 \cdot \overline{x}_0 \\ z_1 = \overline{x_1} \cdot x_0 \\ z_0 = \overline{x_1} \cdot \overline{x_0} \end{cases}$$

cioè ogni "indice" del decoder corrisponde al prodotto dei due ingressi opportunamente negati: l'ultima uscità avra tutti i bit attivi (sarebbe $2^N - 1$ considerando numeri naturali), ergo prende il prodotto di tutti gli ingressi. Di contro, la prima uscita (0) avrà tutti i bit disattivi, quindi prenderà il prodotto di tutti gli ingressi negati. Gli altri numeri vengono indirizzati prendendo il prodotto e complementando i bit che quel particolare numero si aspetterebbe come 0. Notiamo che, sebbene si abbiano 4 negazioni, nella rete fisica conviene negare gli input in entrata risparmiando 2 invertitori.

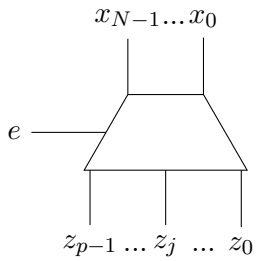
Per le figure, rimandiamo a `https://github.com/Guray00/IngegneriaInformatica/blob/master/SECONDO%20ANNO/I%20SEMESTRE/Reti%20Logiche/Diapositive%20OCR/Reti%20combinatorie%20ocr.pdf`.

Generalizziamo quindi questa struttura a decoder da N a 2^N, applicando quanto detto prima. Si avrà:

$$\begin{cases} z_0 = \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot ... \cdot \overline{x_1} \cdot \overline{x_0} \\ z_1 = \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot ... \cdot \overline{x_1} \cdot x_0 \\ ... \\ z_{p-2} = x_{N-1} \cdot x_{N-2} \cdot ... \cdot x_1 \cdot \overline{x_0} \\ z_{p-1} = x_{N-1} \cdot x_{N-2} \cdot ... \cdot x_1 \cdot x_0 \end{cases}$$

1.1.1 Decoder con enabler

Il problema dei decoder come appena descritti è che non sono espandibili: non si possono costruire, come avevamo visto per i gli AND o gli OR, reti di più decoder combinati. Introduciamo per questo motivo il decoder con **enabler**:



Questi decoder hanno $N + 1$ ingressi, cioè quelli normali più l'enabler, che ha il compito di "accendere" il decoder stesso. Fisicamente, potremmo semplicemente inserire il decoder e come ingresso aggiuntivo agli AND già predisposti, per avere che:

$$z_i = \begin{cases} y_i & e = 1 \\ 0 & e = 0 \end{cases}$$

e quindi:

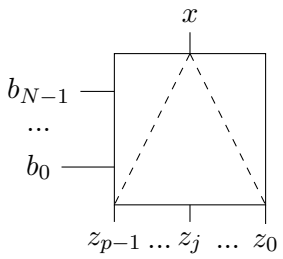
$$\begin{cases} z_0 = e \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot ... \cdot \overline{x_1} \cdot \overline{x_0} \\ z_1 = e \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot ... \cdot \overline{x_1} \cdot x_0 \\ ... \\ z_{p-2} = e \cdot x_{N-1} \cdot x_{N-2} \cdot ... \cdot x_1 \cdot \overline{x_0} \\ z_{p-1} = e \cdot x_{N-1} \cdot x_{N-2} \cdot ... \cdot x_1 \cdot x_0 \end{cases}$$

Adesso basta accorgersi che reti di decoder con $N > 2$ possono crearsi concatenando decoder a decoder, cioè usando un decoder con i bit più significativi in entrata per generare l'enabler di N nuovi decoder, i quali ricevono i bit meno significativi in entrata.

Ad esempio, se vogliamo creare un decoder `4to16` a partire da decoder `2to4`, useremo 4 decoder, con gli stessi input (x_0 e x_1), abilitati da un quinto decoder con input x_2 e x_3.

1.2 Demultiplexer

Il demultiplexer è una rete con $N + 1$ ingressi e $p = 2^N$ uscite:



Chiamiamo x la **variabile da commutare**, e le altre **variabili di comando** (b). La j-esima uscita insegue la variabile da commutare se e solo se:

$$(b_{n-1}, ..., b_0)_2 = j$$

altrimenti vale 0. Questo significa che il demultiplexer invia il suo input, x, all'output z_j tale che i controlli $b_{N-1}...b_0$ sono la codifica binaria di j.

Il multiplexer, fisicamente, è identico ad un decoder con enabler: si fa la parte di decoding con il:

$$\begin{cases} z_0 = \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot ... \cdot \overline{x_1} \cdot \overline{x_0} \\ z_1 = \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot ... \cdot \overline{x_1} \cdot x_0 \\ ... \\ z_{p-2} = x_{N-1} \cdot x_{N-2} \cdot ... \cdot x_1 \cdot \overline{x_0} \\ z_{p-1} = x_{N-1} \cdot x_{N-2} \cdot ... \cdot x_1 \cdot x_0 \end{cases}$$

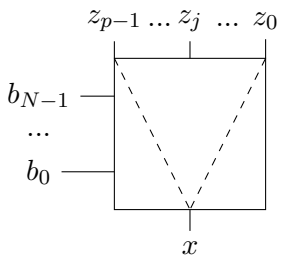
di prima, e si moltiplica per x per ottenere il comportamento desiderato:

$$\begin{cases} z_0 = x \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot ... \cdot \overline{x}_1 \cdot \overline{x_0} \\ z_1 = x \cdot \overline{x_{N-1}} \cdot \overline{x_{N-2}} \cdot ... \cdot \overline{x_1} \cdot x_0 \\ ... \\ z_{p-2} = x \cdot x_{N-1} \cdot x_{N-2} \cdot ... \cdot x_1 \cdot \overline{x_0} \\ z_{p-1} = x \cdot x_{N-1} \cdot x_{N-2} \cdot ... \cdot x_1 \cdot x_0 \end{cases}$$

Con $x = e$ questo è un decoder con enabler x.

1.3 Multiplexer

Il multiplexer è il duale del demultiplexer: una rete con $N + 2^N$ ingressi e 1 uscita:



Gli ingressi b_i si chiamano variabili di comando, e selezionano l'ingresso connesso all'uscita come:

$$z = x_i \Leftrightarrow (b_{N-1}, ..., b_1, b_0) = i$$

Abbiamo detto che il multiplexer è il duale del demultiplexer: se quest'ultimo prendeva un segnale x e lo inviava al j-esimo output sulla base della codifica di j ottenuta alle variabili di controllo, il multiplexer prende il j-esimo ingresso, secondo gli stessi canoni, e lo invia alla linea x di uscita.

Alla base della sintesi di un multiplexer sta un decoder: infatti, abbiamo che quest'ultimo seleziona uno solo (*one-hot*) degli output, che possiamo moltiplicare (mettiamo una AND) per l'ingresso corrispondente. Visto che solo uno degli output in uscita dagli AND è attivo in un dato momento, possiamo ricombinare il segnale finare con un unico grande OR.

Come prima, possiamo eliminare gli AND in cascata dal decoder connettendoli agli AND già contenuti in esso.

Otteniamo quindi la descrizione algebrica (si noti che adesso abbiamo fatto sintesi $\rightarrow$ descrizione, mentre fino a questo punto avevamo fatto l'operazione inversa, descrizione $\rightarrow$ sintesi):

$$\begin{aligned} z = x_0 \cdot \overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdot ... \cdot \overline{b_1} \cdot \overline{b_0} + \\ x_1 \cdot \overline{b_{N-1}} \cdot \overline{b_{N-2}} \cdot ... \cdot \overline{b_1} \cdot b_0 + \\ ... + \\ x_{p-2} \cdot b_{N-1} \cdot b_{N-2} \cdot ... \cdot b_1 \cdot \overline{b_0} + \\ x_{p-1} \cdot b_{N-1} \cdot b_{N-2} \cdot ... \cdot b_1 \cdot b_0 \end{aligned}$$

Notiamo che il multiplexer è una rete a 2 livelli di logica: il segnale passerà al massimo da un AND e un OR. Le NOT sugli ingressi non si contano, in quanto in una rete fisica le variabili di comando proverranno da registri, che forniscono già una versione negata del loro output senza bisogno di ulteriori inversori.

1.3.1 Multiplexer come rete combinatoria universale

Dimostriamo il seguente teorema:

Teorema 1.1: Multiplexer come rete combinatoria universale

Un multiplexer con N variabili di comando è in grado di realizzare qualunque legge combinatoria ad N ingressi ed un uscita, connettendo i 2^N ingressi a generatori di costante.

Abbiamo che:

- Un multiplexer si ricava con porte AND, OR e NOT a due livelli di logica;
- Un multiplexer realizza qualsiasi rete combinatoria ad un'uscita;
- una rete a più uscite può essere scomposta in più reti con le uscite messe "in parellelo".

Allora qualsiasi rete combinatoria può essere creata combinando AND, OR e NOT su due livelli di logica.

Inoltre, si può dimostrare che per qualsiasi tabella di verità ad N ingressi, si può trovare una rete che la implementa tramite un multiplexer a $N-1$ variabili di comando, e al più porte NOT.

1.4 Modello strutturale universale per reti combinatorie

Vediamo adesso un modo per sintetizzare una rete logica ad N ingressi ed M uscite a partire da una tabella di verità. Si prende prima di tutto un decoder con N ingressi, e si creano M linee parallele alle 2^N (che è anche il numero delle righe della tabella di verità) linee di uscita del decoder. Si combinano quindi queste linee di uscita attraverso OR su ogni intersezione che corrisponde ad una certa cella della tabella di verità.

1.4.1 Riduzione dei costi

Definiamo informalmente il costo come ridotto quando si usano meno porte logiche. Troviamo quindi un modo per ridurre il costo della rete creata. Avremo che, inizialmente, tutte le uscite si presentano in una forma canonica **SP**, che sta per Somma di Prodotti, del tipo:

$$z_j = x_{n-1} \cdot ... \cdot x_0 + ... + x_{n-1} \cdot ... \cdot x_0$$

con la possibilità di complementare qualsiasi x. Questa forma equivale effettivamente a una forma normale disgiuntiva.

Possiamo quindi usare le proprietà dell'algebra di Boole per raggruppare e semplificare i termini. Vogliamo un algoritmo che ci permetta di eseguire questi passaggi in modo ordinato, e ci porti sempre alla soluzione ottimale.