# Network Semplification

## MV-network

Una rete multi-valore è una rete dove ogni nodo rappresena una fuznzione multi-valore con un singolo output (sempre mulit-valore).

Chiamiamo questi valori *i-sets*, il set 0 è l’inizio della funzione in cui il nodo ha valore 0. C’è una variabile MV associata all’out-put di ogni nodo.

Le funzioni possono essere collegate direttamente ad altre funzioni con un collegamento creando dipendenza direta tra i due nodi.

Un network ha una serie di input iniziali e uno o più output finali.

Una delle differenze principali deille reti binarie è che i nodi-mv possono avere ognuno il proprio range di output possibili (diverso diminio tra nodi).

## Combinationale Optimization

### Node simplification

L’i-set per ogni nodo della rete può essere semplificato usando vari comandi di semplificazione. Solitamente viene utilizzato un minimizzatore come **ESPRESSO-MV** che:

* Minimizza gli input del nodo (elimina i nodi inutilizzati ?)
* Converte le funzione in funzioni binari.

Questa struttura di semplificazione a 2 livelli seve per trovare una rappresentazione logica con il minor numero implicanti (**cubi**) e di letterali preservando la funzionalità della funzione.

Il nuovo nodo semplificato viene messo nella rete sostituendo la sua controparte non semplificata.

**Fullsimp** utilizza i concetti sopra descritti per trovare una semplificazione del nodo. E’ stata poi implementata una vesrione più performante chimata **fullsimp\_complete** ma i conetti di base restano gli stessi.

### Algebraic MV Methods

Vengono utilizzate le funzioni algebriche per ottenere nuovi nodi che rappresentazno le funzioni dei nodi.

1. **Sostituzione algebrica**: utilizza la divisione algebrica, agisce in 2 modi, in base al tipo di cubo. Se cubo a 2 usa un metodo veloce, quello più lento altrimenti.
2. **Node extraction**: guarda tutti i nodi della rete e cerca di estrarre fattori comuni efficaci e creare nuovi nodi nella rete, riesprimendo altri nodi in termini di questi nodi appena introdotti. È una delle trasformazioni usate per scomporre grandi funzioni in pezzi più piccoli.
3. **Factorization**: crea una forma fattorizzata per ogni nodo.

### Network manipulation

1. **Collapsing**: converte l’intera rete multilivello in modo che le forme SOP per ogni uscita siano in termini di soli ingressi primari. Così il numero di nodi nella rete sarà esattamente il numero di uscite primarie.
2. **Merging**: Prende tutti i nodi e forza un’unione creando un singolo nodo multi-valore costruendo in *i-set* per ogni combinazione di valori creata. Se ci sono più *i-set* creati uguali vengono uniti in un singolo nodo.
3. **Encoding**: è come l’inverso del merging di funzioni binarie. Cerca di trovare una buona codifica binaria per ogni variabile multivalutata nella rete, compresi gli ingressi e le uscite primarie. Alla fine, ogni segnale è codificato come un segnale binario. Quindi un file binario può essere scritto. Formato da 2 fasi:
   1. Inizia dagli ingressi e per ogni nodo, determina se uno dei suoi fanin può essere usato per codificare parzialmente il nodo.
   2. inizia dalle uscite e in ordine topologico inverso lavora a ritroso fino agli ingressi primari. Ad ogni nodo, le sue uscite sono codificate utilizzando le informazioni su come sono utilizzati i suoi fanout
4. **Pair decoding**: Simile al merging ma utilizza un altro modo per scegliere quali modi unire.
5. **Bi-decomposition**: crea dei nodi multi-valore intermedi. Prende una rete MV appiattita o parzialmente appiattita e ne genera un’altra composta da porte MAX e MIN multi-valutate a due ingressi e da iterali multi-valutati. Vengono sfruttate sia l’incompletezza della specifica iniziale che le flessibilità generate nel processo di ecomposizione.

## comandi di sintesi all’interno di MVSIS

* **sweep**: semplifica il circuito rimuovendo i nodi inutili
* **strash**: sostituisce il circuito corrente con un circuito equivalente di AND/INV
* simplify
* **mfsn**: semplifica il circuito utililzzando i dont’t care
* **merge** : unisce tutti i nodi della lista
* **lxu**: performs layout-aware unate fast extract on the current network
* **fxu**: performs unate fast extract on the current network
* **fullsimp**: sintesi utilizzata per la sintesi finale del circuito, ha 6 tipi diversi di algoritmi di sintesi che si possono scegliere
* **fraig\_sweep**: analizza i nodi funzionalmente equivalenti utilizzando i FRAIG
* **fraig**: trasforma la rete corrente in una rete di AND
* **fd**: esplora la dipendenza dei nodi della rete attuale usando SAT
* **espresso**: minimizza ogni nodo utlizzando espresso
* **encode**: nessuna descrizione nella documentazione
* **eliminate**: fa collassare selettivamente i nodi nei loro fanout
* **dsd**: solo con biari, inutile per noi
* **decomp**: decomprime la rete utilizzando le forme fattorizzate
* **collapse2/collapse**: crea una rete a un solo livello
* **club**: raggruppa i nodi in cluster e li fa collassare in un unico livello (i cluster credo)
* **balance**: sostituisce la rete attuale con una rete di AND2s ben formata
* **add\_buffer**: aggiunge un buffer dopo ogni CI e prima di ogni CO