Il linguaggio Verilog HDL

Introduzione al linguaggio Verilog HDL per la descrizione di sistemi digitali

TESTI Consigliati

- J. Bhasker
  Verilog HDL Synthesis – A pratical Primer
  Star Galaxy Publisher

Obiettivi

- NON è un corso completo sul Verilog
- NON imparerete nel dettaglio la sintassi del Verilog
- NON si approfondiranno gli aspetti semantici
- NON imparerete a sviluppare estensioni o integrazioni

- Capirete quali sono i motivi che spingono ad impiegare il Verilog
- Comprenderete i concetti base e sarete in grado di interpretare un sistema descritto tramite Verilog
- Sarete in grado di descrivere sistemi di “utilità pratica”in Verilog

Introduzione

A chi e' dedicato?

Ai progettisti di circuiti e sistemi logici che utilizzano sistemi CAE (Computer Automated tools for Electronic design)

Un po' di storia:

- Sviluppato inizialmente nel 1984-85 da Philip Moorby
- Nel 1990 Cadence lo adotta sotto il nome di Verilog-XL
- Standardizzazione IEEE arriva nel 1995 seguita da un'integrazione nel 2001
- Viene successivamente esteso nella versione System-Verilog (più orientato alla descrizione di sistemi)
- Porta alla nascita di Verilog-A (per sistemi analogici)

Attualmente: A causa della attuale complessità dei circuiti ha surclassato altre metodologie (schematic capture)

ALTERA

Cos'è (e cosa NON è) Verilog HDL

- HDL = Hardware Description Language
- **NON** è un linguaggio di programmazione SW
  - Un linguaggio di programmazione software descrive una serie di operazioni che vengono eseguite secondo una determinata sequenza su di un processore dedicato.
- E' un linguaggio di descrizione Hardware
  - Descrive il funzionamento e/o la struttura di un circuito logico che è già stato, oppure sarà realizzato con un HW dedicato
  - Descrive eventualmente il susseguirsi di segnali logici con cui sollecitare il circuito

Obiettivi

- DOCUMENTARE
  - Ad esempio il sistema è già stato realizzato e si vuole valutare la compatibilità dei segnali verso altri sistemi
- SIMULARE
  - Si possono generare i segnali di stimolo atti a simulare il funzionamento di un sistema.
- SINTETIZZARE
  - Descrivere il sistema a diversi livelli di astrazione via via più precisi, fino a giungere a quello più basso (es. porte logiche) atto ad essere realizzato fisicamente

Vantaggi (rispetto altre tipologie di descrizione)

- Progetto di tipo “technology – independent”
  - Impiegato da diversi fornitori / venditori (sia di HW che SW)
  - Facilita gli “aggiornamenti” del sistema progettato
  - La documentazione del progetto è “STANDARD”
- Miglioramento nella qualità del progetto
  - Consente di analizzare varie alternative
  - Consente più livelli di astrazione
    - Verifica ad un elevato livello di astrazione
    - Paragone delle prestazioni tra vari livelli di astrazione
    - Integrazione tra blocchi sviluppati a vari livelli
  - Riutilizzo e condivisione di blocchi già sviluppati

Considerazioni

- E’ meno “immediato” di uno schema digitale
- E’ sintatticamente … più pesante
- Non tutto ciò che viene scritto in Verilog è fisicamente “sintetizzabile”

- Esistono tools di sviluppo che consentono di ricavare descrizioni in Verilog partendo da Schemi, Macchine a Stati Finiti, Tabelle di verità, grafi, diagrammi temporali ecc….

Concetti fondamentali

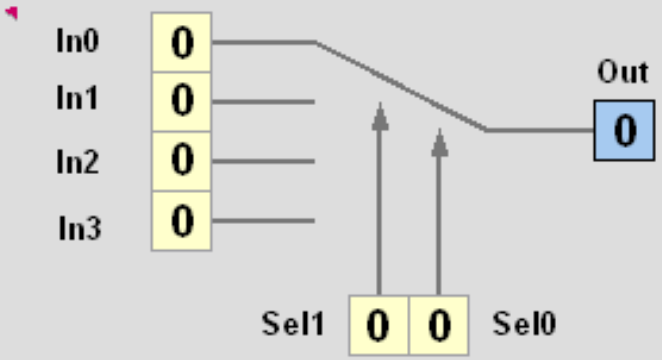
- Descrizione di circuiti LOGICI
- Descrizione secondo diversi livelli di astrazione
  - Tra cui la possibilità di descrivere una struttura
- Possibilità di descrivere la contemporaneità nell'esecuzione dei processi
- Possibilità di descrivere particolari "stati" di un collegamento fisico (0, 1, z, x)

Livelli di astrazione

- Non sempre un sistema deve essere definito con un elevato grado di dettaglio
  - Potrebbe non essere stato ancora realizzato
  - Potrebbe essere un modo per testarne le specifiche e/o la compatibilità tra sistemi interagenti
  - Potrebbe essere un modo per “simulare” il funzionamento di un elemento già esistente
- Un basso livello di astrazione
  - È utile quando si voglia realizzare fisicamente il dispositivo
  - Tipicamente esce da un processi di sintesi (automatica o manuale)

Livello Comportamentale

- Descrive il funzionamento del sistema senza esplicitare i meccanismi che lo realizzano



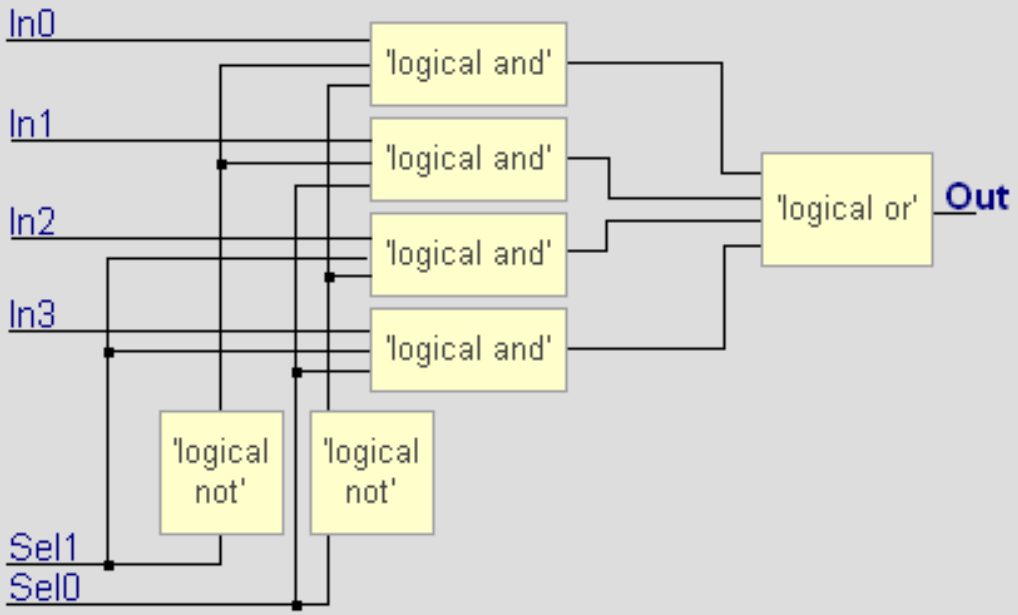

```
module mux_4_to_1 (Out,In0,In1,In2,In3,Sel1,Sel0);
output Out;
input In0, In1, In2, In3, Sel0, Sel1;
reg Out;

always @(Sel1 or Sel0 or In0 or In1 or In2 or In3)
  begin
    case ({Sel1, Sel0})
    2'b00 : Out = In0;
    2'b01 : Out = In1;
    2'b10 : Out = In2;
    2'b11 : Out = In3;
    default : Out = 1'bx;
    endcase
  end

endmodule
```

Descrizione Comportamentale

- Descrive il funzionamento del sistema NON la sua struttura fisica
- Non vi è un intento specifico di descrivere dell'HW, infatti potrebbe essere usato per
  - Descrivere le specifiche di un sistema da realizzare
  - Descrivere un flusso di segnali (es.per il simulatore)
  - Descrivere un sistema già esistente senza entrare nel dettaglio (es. per verificare la compatibilità)
- E' una descrizione assolutamente generica e non focalizzata ad un determinato "target"
- Non tutto ciò che viene descritto in tal modo può (o deve) essere oggetto di sintesi

Livello “dataflow”

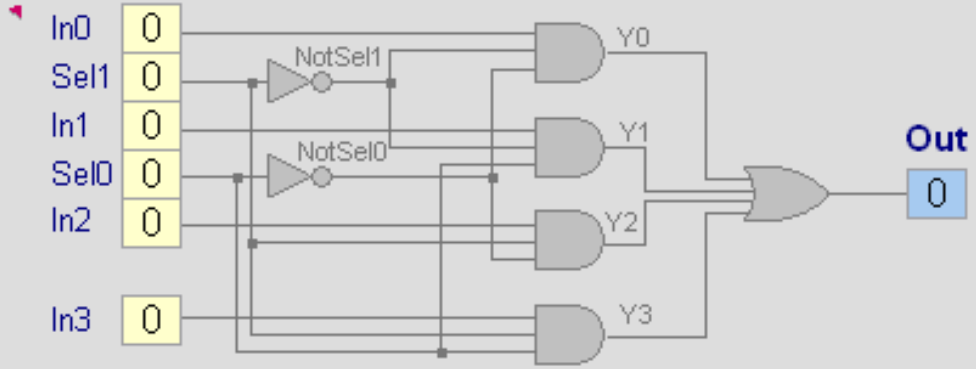
- Rappresenta il funzionamento in base al “flusso di dati” che viene elaborato




```
module mux_4_to_1 (Out, In0, In1, In2, In3, Sel1, Sel0);

output Out;
input In0, In1, In2, In3, Sel0, Sel1;

assign Out = (~Sel1 & ~Sel0 & In0) | (~Sel1 & Sel0 & In1)
             | (Sel1 & ~Sel0 & In2)| (Sel1 & Sel0 & In3);

endmodule
```

Livello Strutturale

- Rappresenta una struttura composta da più blocchi funzionali.




```
module mux_4_to_1 (Out,In0,In1,In2,In3,Sel1,Sel0);
output Out;
input In0, In1, In2, In3, Sel0, Sel1;

wire NotSel0, NotSel1;
wire Y0, Y1, Y2, Y3;

not (NotSel0, Sel0);
not (NotSel1, Sel1);
and (Y0, In0, NotSel1, NotSel0);
and (Y1, In1, NotSel1, Sel0);
and (Y2, In2, Sel1, NotSel0);
and (Y3, In3, Sel1, Sel0);
or (Out, Y0, Y1, Y2, Y3);

endmodule
```

Descrizione Strutturale

- Descrive la struttura di un circuito come interconnessione di elementi
- Se si vuole poterla simulare, ogni singolo elemento che la compone deve essere “noto”
- Spesso è la descrizione che viene generata al completamento di un processo di sintesi
- Spesso è dedicata ad un HW specifico e pertanto può risultare meno “target independent”.

Contemporaneità dei processi

- Nel mondo reale (e nei circuiti logici) i processi sono tra loro “concorrenti”
- I linguaggi di descrizione SW descrivono una serie di operazioni da svolgere in sequenza
- In Verilog HDL si possono descrivere sia operazioni tra loro concorrenti che in forma sequenziale
- Simulazioni in base al “tempo simulato”