UE RECHNERARCHITEKTUR 1

SystemVerilog

Alwin Zulehner

28. April 2016

JOHANNES KEPLER
UNIVERSITÄT LINZ

Signale

Signale entsprechen den Drähten in einer digitalen Schaltung und werden in SystemVerilog mit dem Keyword `logic` angelegt. Sie dienen als Ein- und Ausgänge von Schaltkreisen und können zur Berechnung von Zwischenergebnissen verwendet werden.

- Signal `a` mit einer Breite von einem Bit:
  ```
  logic a;
  ```
- Signal `b` mit einer Breite von 4 Bit:
  ```
  logic [3:0]b;
  ```
- Gruppieren von Signalen mit geschwungenen Klammern:
  ```
  { a,b }
  ```

Zuweisungen

Einem Signal kann man mit dem Keyword `assign` eine Boolsche Funktion oder eine Konstante zuweisen. Es können auch mehrere Bits eines Signals auf einmal gesetzt werden.

- `assign a = b & c;`
- `assign b[1] = 1'b0;`
- `assign b[3:2] = {a, a};`
- `{a, a} = 2'b11;`

NOT	$\sim$
AND	&
OR	|
XOR	^

Konstanten habe die Form `<width>'<base><const>`, wobei `<width>` die Bitbreite angibt, `<base>` die Basis (**b**inär, **d**ezimal oder **h**exadizimal) und `<const>` den konstanten Wert.

Module

Schaltkreise werden in SystemVerilog als Module bezeichnet:

- Multiplexer $y = \overline{sel} \cdot a + sel \cdot b$

```
module mux(a,b,sel,y);
    input a,b,sel;
    output y;

    assign y = ~sel & a | sel & b;
endmodule
```

Instantiieren eines Moduls:

- `mux m1(c,d,s,z);`

Testbench

Mit einer testbench wird das Verhalten der Schaltung getestet (z.B.: gegen eine verhaltensbasierte Beschreibung)

- Der `initial` Block wird zu Beginn der Simulation ausgeführt.
- Um eine Zeiteinheit in der Simulation zu warten wird der Befehl `#1;` verwendet. Das ist nötig um mehrere Zuweisungen hintereinander zu machen.
- Mit Schleifen können alle möglichen Belegungen der Inputs durchgegangen werden.
- Mit dem Befehl `$monitor(...)` werden alle Änderungen der Variablen an der Konsole ausgegeben.
- Mit dem Befehl `$display(...)` kann Text an der Konsole ausgegeben werden.

Strukturelle vs. verhaltensbasierte Beschreibung

Mit struktureller Beschreibung erhält man ein synthetisierbares Design. Verhaltensbasierte Beschreibung wird meist zum Testen verwendet und ist nicht direkt synthetisierbar.

Strukturell

- Boolsche Operatoren ($\sim$, &, |, ^)
- Module, Signale

Verhaltensbasiert

- Arithmetische Operationen (+,-,*,/)
- Kontrollstrukturen `if, for, initial, ...`

Beispiel: Addierer I

```
module FA(a, b, cin, s);
    input a,b,cin;
    output [1:0]s;

    assign s[0] = a ^ b ^ cin;
    assign s[1] = a & cin | b & a | b & cin
        ;
endmodule

module testbench;
    logic [3:0] a,b;
    logic [3:0] sum, sum2;
    logic c1, c2;

    FA f1(a[0], b[0], 1'b0, {c1, sum[0]});
    FA f2(a[1], b[1], c1, {c2, sum[1]});
    FA f3(a[2], b[2], c2, sum[3:2]);

    assign sum2 = a+b;
    integer errors;
```

Beispiel: Addierer II

```
    initial begin
        $monitor($time, ": a=%b (%d) b=%b
            (%d) sum=%b (%d)", a,a,b,b,sum,
            sum);
        errors = 0;
        for(a=4'b0; a <= 4'b0111; a++)
            for(b=4'b0; b < 4'b1000; b++)
                begin
                 #1;
                 if(sum2 != sum) begin
                     errors++;
                 end
            end

        $display("Errors: %d", errors);
    end
endmodule
```

Simulation

Simulation mit ModelSim-Altera

- Download unter
  `http://dl.altera.com/?edition=standard&product=modelsim_ae#tabs-2`
- Ausführen unter Linux:
  `<Install Dir>/altera/15.1/modelsim_ase/bin/vsim`
- Im Fehlerfall `Error: cannot find ./../linux_rh60/vsim`

```
cd <Install Dir>/altera/15.1/modelsim_ase/
user $ ln -s ./linuxaloem ./linux_rh60
```

Simulation

Simulation in EPIIC:

1. Datei im Workspace auswählen
2. Als Programm Modelsim wählen
3. Auf `run` klicken

In EPIIC können Textdateien auch bearbeitet werden (Klick auf die Datei und danach links auf `Edit`)

Bevor SystemVerilog Dateien abgegeben werden können müssen diese mit ModelSim ausgeführt werden. Dateien, die nicht kompilieren können nicht abgegeben werden.