HDL Synthesis Idioms

This lecture describes the common idioms understood by logic synthesizers and how they are converted to circuits. After this lecture you should be able to convert back and forth between digital logic circuits and the corresponding System Verilog descriptions.

Introduction

Hardware Description Languages are used for both simulation and synthesis.

Logic synthesizers (such as Synopsys' Synplicity, Intel's Quartus and Xilinx's Vivado) convert HDL descriptions into circuits (actually, netlists). They do this by recognizing a relatively small number of idioms.

In order to be able to generate efficient designs, a designer must be able to visualize the hardware that would be generated by an HDL description. This lecture describes some common HDL constructs and their corresponding hardware implementations.

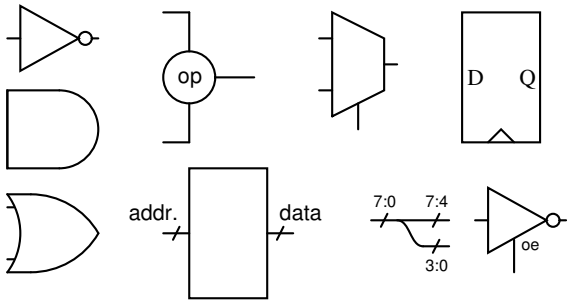
There are minor variations between tools which are described in each tool's documentation.

Combinational Logic

Although the synthesizer is free to generate any hardware that has the specified functionality, the following HDL constructs are typically synthesized as follows:

- logical operators are converted to the equivalent logic gates as you would expect.
- arithmetic and comparison operators are converted to (usually optimized) blocks of combinational logic that implement the required operation.
- access to a value in an (unpacked) array is implemented as a read-only memory (ROM). Access to a packed array is implemented as a connection to specific bits of a bus.
- ternary operators, if/else and case statements are converted into multiplexers; nested as necessary.
- an output to which 'z' can be assigned is converted to a tri-state output.

Exercise 1: Using the schematic symbols shown below, convert each of the following System Verilog expressions into a schematic.




```
y = a ^ b ;

y = a < b ;

y = y+1 ;

y = a[3] ;

y = a[3] ? 4 : a[2] ? 3 : a[1] ? 2 :
    a[0] ? 1 : 0;

y = table[x] ;

if ( y < b )
  y = y+1 ;
else
  y = y-1 ;

y = oe ? d : 16'hzzzz ;
```

Sequential Logic

The following HDL constructs are synthesized as follows:

- `always` blocks with sensitivity lists containing signal edges are converted to flip-flops or registers. The signals assigned to within the `always` block are the register outputs.

- combinational logic that computes the values assigned to register outputs are used to describe specialized sequential logic such as counters and shift registers. The register's current value is often used to compute the next value.

Exercise 2: Using the schematic symbols shown above, convert each of the following System Verilog expressions into a schematic.

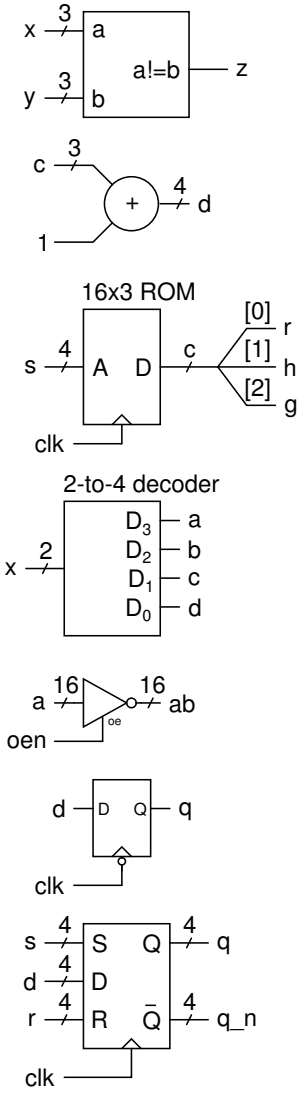
```
always@(posedge clk)
  y = a ;

always@(posedge clk)
  y[7:0] <= {y[6:0],a} ;

always@(posedge clk)
  if ( e )
     y <= a ;
  else
     y <= y ;

always@(posedge clk)
  if ( r )
    y <= '0 ;
  else
    if ( e )
       y <= y+1'b1 ;
    else
       y <= y ;

next = ( reset || done ) ? '0 : cnt+'b1 ;

always@(posedge clk)
  if ( falling )
     mosi <= sr[31] ;

always@(posedge clk)
   cnt <= cnt_next ;

// logic [31:0] mem [15:0]
always_ff@(posedge clk) begin
   mem[p] <= din ;

// logic [31:0] mem [15:0]
dout = mem[p] ;

p_next = valid && rdy ? p + 1'b1 : p  ;

// i, j are logic[4:0]; w, sclk are logic
nxt = w ? 5'd7 : ( j==N && sclk ) ? i-1 : i ;

readdata = {31'b0,csn} ; // csn is logic

crc = ^ (g&sr) ; // g and sr are logic[31:0]

nxt = ~d[8] ;
```

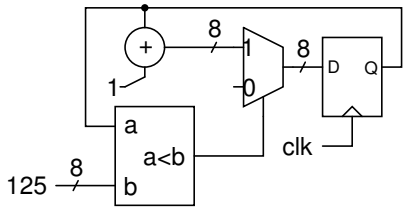
Schematics to HDL

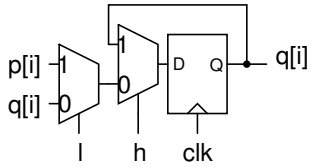
It's also important to be able to write the HDL that will result in a specific hardware architecture.

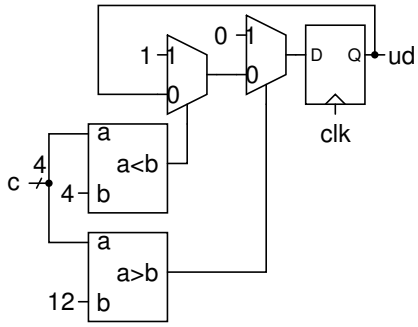
Each circuit element is converted into the corresponding HDL construct and named signals are used to connect them according to the circuit topology.

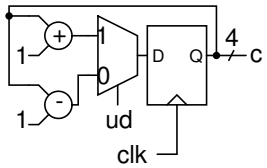
Exercise 3: Write System Verilog that would generate each of the following schematics. Include any required signal declarations (using `logic`).

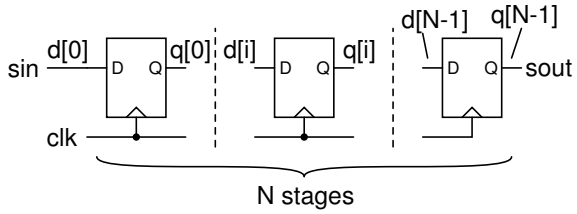



+
8
1
1
0
8
D Q
clk
a
a<b
b
125
8


p[i]
q[i]
1
0
1
0
D Q
q[i]
l
h
clk


1
1
0
0
1
0
D Q
ud
clk
a
a<b
c
4
4
b
a
a>b
12
b


+
1
1
0
D Q
4
c
-
1
ud
clk


d[N-1]
q[N-1]
sin
d[0]
D Q
q[0]
d[i]
D Q
q[i]
D Q
sout
clk
N stages