Prof. Bernd Finkbeiner
Dr. Calogero Zarba
Moritz Hahn

Sample Solution for Problem Set 5
Due: June 12, 2007, 9:14 am

Embedded Systems

Problem 1 (VHDL)

Notice that original specification of h is equivalent to

$$h = ((\neg x) \wedge a) \vee (x \wedge b)$$

Behavioral specification

We can directly transform the modified specification of h to a behavioral VHDL specification:

```
architecture behavior of multiplexer is
begin
  h <= ((not x) and a) or (x and b);
end behavior
```

Structural specification

Consider figure 1. We can transform this circuit into a structural specification:

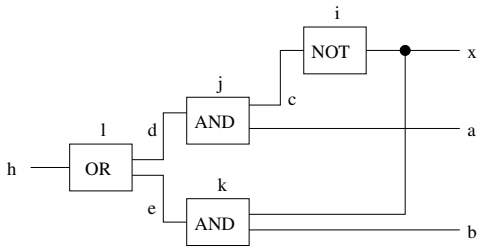

Figure 1: Circuit for the multiplexer

```
architecture structural of multiplexer is
signal c, d, e: Bit
begin
  i: NOT port map(x, c);
  j: AND port map(c, a, d);
  k: AND port map(x, b, e);
  l: OR port map(d, e, h);
end structure
```

Here we assume that `NOT`, `AND` and `OR` have their input parameters before their output parameters in order.

Problem 2 (VHDL)

Adder without saturation

Consider figure 2. It implements an adder without saturation and we can easily implement it as both a behavioral and a structural VHDL model.

Behavioral specification

```
architecture behavior of adder is
variable e, f, g: Bit;
begin
  e := a0 and b0;
  f := (a1 and b1) or (a1 and e) or (a2 and e);
  g := (a2 and b2) or (a2 and f) or (b2 and f);
```

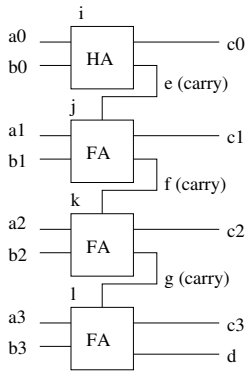

Figure 2: Circuit for the adder without saturation

```
  c0 <= a0 xor b0;
  c1 <= a1 xor b1 xor e;
  c2 <= a2 xor b2 xor f;
  c3 <= a3 xor b3 xor g;
  d <= (a3 and b3) or (a3 and g) or (b3 and g);
end behavior
```

Notice that we have used variables here, an opportunity not given in the lecture. You can, however, replace recursively each instance of `e`, `f` and `g` in the assignments to `c0`, `c1`, `c2`, `c3` and `d` to get a description without variables. But this would not add to clarity of the solution probably.

Structural specification

```
architecture structure of adder is
signal e, f, g: Bit
begin
  i: HA port map(a0, b0, c0, e);
  j: FA port map(a1, b1, e, c1, f);
  k :FA port map(a2, b2, f, c2, g);
  l: FA port map(a3, b3, g, c3, d);
end structure
```

Adder with saturation

We can easily change the adder without saturation to one with saturation, as shown in figure 3. We know that we have to saturate if `d` is 1, because then we have an overflow. We do the saturation by `OR`-ing all the previous outputs with `d`.

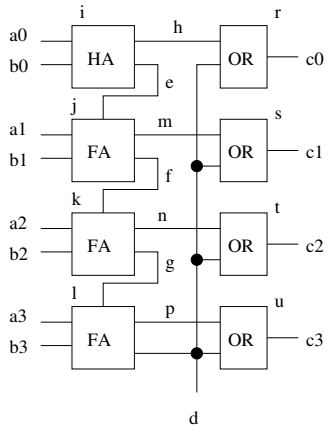

Figure 3: Circuit for the adder with saturation

Behavioral specification

```
architecture behavior of adder_sat is
```

```
variable e, f, g, q: Bit;
begin
  e := a0 and b0;
  f := (a1 and b1) or (a1 and e) or (a2 and e);
  g := (a2 and b2) or (a2 and f) or (b2 and f);
  q := (a3 and b3) or (a3 and g) or (b3 and g);
  c0 <= (a0 xor b0) or q;
  c1 <= (a1 xor b1 xor e) or q;
  c2 <= (a2 xor b2 xor f) or q;
  c3 <= (a3 xor b3 xor g) or q;
  d <= q;
end behavior
```

Structural specification

```
architecture structure of adder_sat is
signal e, f, g, h, m, n, p: Bit
begin
  i: HA port map(a0, b0, h, e);
  j: FA port map(a1, b1, e, m, f);
  k :FA port map(a2, b2, f, n, g);
  l: FA port map(a3, b3, g, p, d);
  r: OR port map(h, d, c0);
  s: OR port map(m, d, c1);
  t: OR port map(n, d, c2);
  u: OR port map(p, d, c3);
end structure
```

Problem 3 (VHDL)

The tabular for the school method for multiplying two binary numbers looks like this:

$$
\begin{array}{lccccccccc}
 & (& 0 & 0 & 0 & a_3 & a_2 & a_1 & a_0 &) \wedge b_0 \\
+ & (& 0 & 0 & a_3 & a_2 & a_1 & a_0 & 0 &) \wedge b_1 \\
+ & (& 0 & a_3 & a_2 & a_1 & a_0 & 0 & 0 &) \wedge b_2 \\
+ & (& a_3 & a_2 & a_1 & a_0 & 0 & 0 & 0 &) \wedge b_3 \\
+ & \text{the carries} & & & & & & & & \\
\hline
\text{Sum} = c_6c_5c_4c_3c_2c_1c_0 & & & & & & & & &
\end{array}
$$

However, we only need $c_0, \ldots, c_3$ and the information whether any of $c_4, \ldots, c_6$ is 1.

From this, we can derive a circuit for the multiplier as shown in figure 4. Now, from this figure, one could construct the behavioral and structural specification like in the examples before. We do not give it here because the construction is the same as in the previous exercise but just way longer. Saturation can be done analogously to the adder.

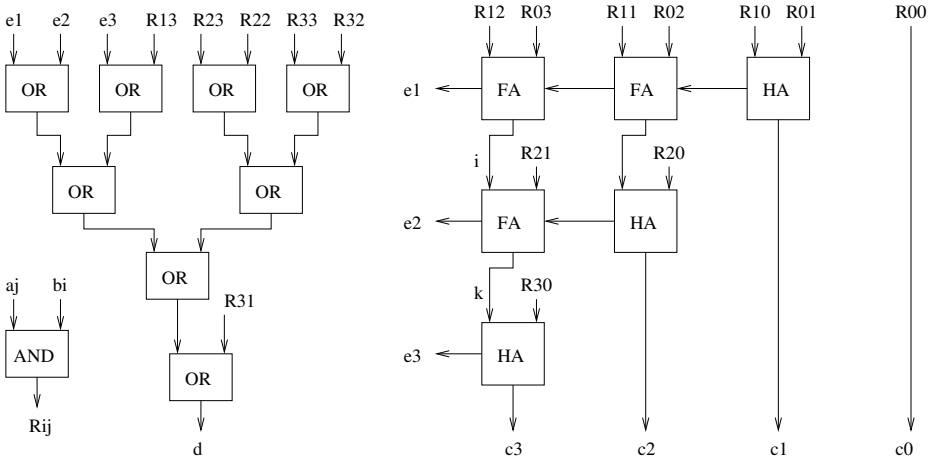

Figure 4: Circuit for the multiplier

The idea is based on `http://fp.cse.wustl.edu/cse362/Lecture%20Notes/rrulecture1.pdf`.