EE789 Mid-semester assignment

Madhav P. Desai

February 6, 2025

The assignment will be done using VHDL. For a basic introduction to VHDL, you may use the following tutorial as a guide.

`https://www.nandland.com/vhdl/tutorials/tutorial-introduction-to-vhdl-for-beginners.html`

1 Shift and Add 8-bit Multiplier

The following pseudo-code claims to implement an 8-bit multiplier (unsigned numbers). It is *not guaranteed* to be correct.

```
// Shift and add multiplier.
//   Uses a single 9-bit adder.
//   Does an 8-bit multiply in
//   1+8 clock cycles.
//
// input start, a[7:0], b[7:0]
// output done, p[15:0]
//
// register ta[7:0]
// register t[16:0]
// registers counter[3:0]
//
// Default outs:
//    done=0, p= 0
rst:
     if start then
```

```
        t[16:0] := 0
        counter := 0
        ta := a
        goto loop
     else
        goto rst
     endif
loop:
     if(ta[0]) then
        // 9-bit adder.
        // (note: for the carry)
        t[16:8] := ((0 & t[16:9]) + (0 & b))
        // shift into lower byte.
        t[7:0]  := t[8:1]
     else
        // shift right.
        t[16:0] := (0 & t[16:1])
     endif
     // shift ta to the right.
     ta[7:0] := (0 & ta[7:1])
     if (counter == 8) then
         goto done_state
     else
         counter := (counter + 1)
         goto loop
     endif
done_state:
     done = 1
     // t is visible at p
     p = t[15:0]
     if start then
        t[16:0] := 0
        counter := 0
        ta := a
        goto loop
     else
        goto done_state
     endif
```

1.1 Assignment

- Confirm that the multiplier works correctly for the case when the input operands are powers of 2.
- Convert the pseudo-code given above to a VHDL description.
- Write a test-bench which checks the results of the multiplier over all 2^{16} input combinations.
  - If the adder does not work correctly, fix it.

2 A faster shift and add 8-bit Multiplier

Now build an 8-bit multiplier which uses four 4-bit multipliers in parallel. The 8-bit inputs a and b are to be viewed as

$$\begin{aligned} a &= a_L + (2^4 \times a_H) \\ b &= b_L + (2^4 \times b_H) \end{aligned}$$

where a_L, a_H, b_L, b_H are 4-bit numbers.

2.1 Assignment

- Use the fork-join construct to write RTL pseudo-code which invokes four parallel 4-bit multipliers and combines their results into a 16-bit final result using 8-bit adders (with carry).
- Convert the RTL pseudo-code to VHDL.
- Use the same test-bench as above to check the results of the multiplier over all 2^{16} input combinations.

3 A Divider!

Build a shift and subtract divider.

3.1 Assignment

- Use a shift-and-subtract algorithm to design an RTL machine (pseudo-code) which implements the division a/b, where a, b are 8-bit numbers.
- Convert the RTL pseudo-code to VHDL.
- Use a similar test-bench as above to check the results of the divider over all 2^{16} input combinations.

4 A Square-root Unit

Build a circuit which calculates the square root of an 8-bit number. That is, given x, it computes the largest possible y such that $y^2 \leq x$.

4.1 Assignment

- Devise an RTL machine (pseudo-code) which implements the square-root function. You may use the components that you have already designed (for example, an 8-bit multiplier).
- Convert the RTL pseudo-code to VHDL.
- Use a similar test-bench as above to check the results of the divider over all 2^{16} input combinations.