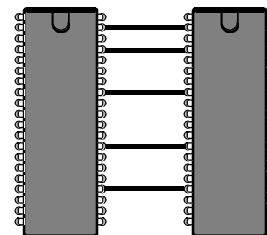


PIN ASSIGNMENTS AND VHDL

In the PLD programming languages, you could assign pins to signals (in the ABEL version we had in the lab, you had to since it didn't have a fitter). So how do we assign pins in VHDL? How do we say that we want signal CLOCK to clock the flip-flops or CLEAR to asynchronously clear them? Actually, we are looking at two different questions, external pin assignments and internal component connections. These questions are often related because in some devices a certain pin is preassigned as the clock pin, and so on.

First, is it necessary or desirable to specify pin numbers? Well, it may be possible to simplify PC board layouts, as shown to the right, by making the right pin assignments. Even with the ability to specify pins, the device structure may not allow this. So, while specifying pin assignments in some cases might be desirable, it is not often very useful.



Can it be done in VHDL? Yes and no! There is a “pin_assignment” attribute which can be used for this purpose. Whether the VHDL compiler you use pays any attention to it or not is another question. The WARP software ignores it! Some software may use it where possible. With small PLD's the ability to assign pins is more useful than with large CPLD's and FPGA's.

Now, regarding internal assignments. If you wanted signal CLOCK to clock flip-flop X and signal CLEAR to asynchronously reset it, you could write (in ABEL),

```
X.clk = CLOCK;  
X.ar = CLEAR;
```

VHDL does not provide this direct ability. VHDL is supposed to be generic. Not all devices may have asynchronous resets, etc. In VHDL these assignments are implied in the specification. Let's look at some examples.

```
if (clock'event and clock= '1') then . . .
```

implies that “clock” is connected to a RET flip-flop. If you use this to program a 22V10, the “clock” signal will be assigned to pin 1 (the clock input on a 22V10). In a CPLD it will be assigned to one of the several clock inputs.

```
if (clock'event and clock= '0') then . . .
```

implies that “clock” is connected to a FET (Falling Edge Triggered) flip-flop. If you compile this into a 22V10, the fitter will generate an error message because a 22V10 has only RET flip-flops. On CPLD's the clock polarity is usually programmable so no error will result.

```
if clear= '1' then x< = '0'; . . .
```

implies an asynchronous reset provided that it is not inside a process involving a clock.

With respect to asynchronous sets and resets, consider the following:

```
process(a,c)
begin
  if a= '1' then f< = '0';
  elsif (c'event and c= '1') then f< = d; end if;
end process;
```

This implies that flip-flop f is to be asynchronously reset. If programmed into a 22V10, the result is:

```
f.D = d
f.AR = a
f.SP = GND
f.C = c
```

If you change the description slightly,

```
process(a,c)
begin
  if a= '1' then f< = '1';
  elsif (c'event and c= '1') then f< = d; end if;
end process;
```

You are implying an asynchronous preset. The 22V10 has none. What happens?

```
/f.D = /d
f.AR = a
f.SP = GND
f.C = c
```

It inverts X and uses the asynchronous reset. If you don't want f to be inverted, then what? You can always instruct WARP2 to maintain the polarities you specify. In this case the result is

```
f.D = d
f.AP = a
f.AR = GND
f.SP = GND
f.C = c
```

Completed Successfully

DESIGN RULE CHECK (13:36:01)

Messages: Error: Asynchronous Preset equation not allowed for f.

In this case an error occurs because there is no asynchronous preset and the design is not fitted. The moral of this story is that you have to know what resources are and are not available on the device you are programming.

As a side note, the 22V10 has a synchronous preset which (to my knowledge) is inaccessible in WARP2.