7.8 Pitfalls and Fallacies

Targeting a bleeding-edge process

There is a fine balance when deciding whether to move to a new process for a new design. On one hand, one is tempted by increased density and speed. On the other hand, support for the new process can initially be quite high (becoming familiar with process rules, CAD tool scripts, porting analog and RF designs, locating logic libraries etc.). In addition, CMOS foundries frequently tune their processes in the first few months of production and frequently yield improvement steps can reflect back to design rule changes that can impact designs late in their tapeout schedule. For this reason, it is frequently prudent not to jump immediately into a new process when it becomes available. On the other hand if you are limited in speed or some other attribute, then you don't have much choice.

Using design rules without understanding their rationale

When designing custom circuits it is important to understand all of the design rules and the reasons for them. Design rules are a compromise between adequate circuit performance while trying to attain the lowest manufacturing cost. Frequently, yield, performance and reliability can be improved without increasing area. A typical example might occur in the selection of power and ground wire widths. The first target is to satisfy the limits. The next target is to satisfy supply rail bounce. The second target is a moving target and it is prudent to make power/ground busses as large as possible consistent with keeping the circuit size down. However, they are not very useful for designs for deep sub-micron processes.

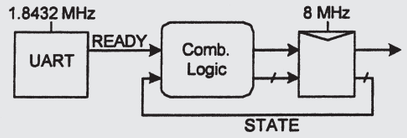

FIG 7.81 Unsynchronized input

Failing to account for the parasitic effects of metal fill

With area density rules, particularly in metal, most design flows include an automatic fill step to achieve the correct metal density. Particularly in analog and RF circuits it is important to either exclude the automatic fill operation from that area or check circuit performance after the fill by completing a full parasitic extract and re-running the verification simulation scripts.

Failing to include process calibration test structures

It was mentioned in the discussion on scribe line structures that test structures are frequently inserted here by the silicon manufacturer. These are frequently undocumented. It is prudent for the designer or design team (particularly in academic designs which are less supported from a foundry) to include their own test structures . This allows one to ascertain where the particular chip lies in the process spread.

However, hierarchy alone does not necessarily solve the complexity problem. For instance, we could repeatedly divide the hierarchy of a design into different submodules but still end up with a large number of different submodules.

$$\pi \cdot 10^{7} = \frac{T_0 e}{(5 \cdot 10^{7})(15 \cdot 10^{-12})} \qquad \textbf{(1.33)}$$

With regularity as a guide, the designer attempts to divide the hierarchy into a set of similar building blocks. The use of iteration to form arrays is an illustration of the use of regularity in an IC design.pendent place. The growth in transistor counts has come from

Table 7.2 MIPS instruction set (subset supported)

Instruction	Function		Encoding	op	funct
`add $1, $2, $3`	`addition:`	`$1 ? $2 + $3`	R	000000	100000
`sub $1, $2, $3`	`subtraction:`	`$1 ? $2 - $3`	R	000000	100010
`and $1, $2, $3`	`bitwise and:`	`$1 ? $2 and $3`	R	000000	100100
`or $1, $2, $3`	`bitwise or:`	`$1 ? $2 or $3`	R	000000	100101
`slt $1, $2, $3`	`set less than:`	`$1 ? 1 if $2 < $3` `$1 ? 0 otherwise`	R	000000	101010
`addi $1, $2, $3`	`add immediate:`	`$1 ? $2 + imm`	R	001000	n/a
`beq $1, $2, $3`	`branch equal:`	`PC ? PC + imm`[5] `if $1 = $2`	R	000100	n/a
`j destination`	`jump:`	`PC ? destination`[5]	R	000010	n/a
`lb $1, $2, (imm)`	`load byte:`	`$1 ? mem($2 + imm)`	R	100000	n/a
Total	`store byte:`	`mem($2 + imm) ? $1`	R	110000	n/a

Regularity can exist at all levels of the design hierarchy. At the circuit level, uniformly sized transistors might be used, while at the gate level a finite library of fixed height, variable length logic gates may be used.

- datapath operators
- memory elements
- control structures
  - I/O
  - power distribution
  - clock generation & distribution

At high levels, identical processor structures may be used to achieve performance. Regularity may also aid in verification by eliminating unnecessary simulation or allowing formal verification programs to operate more efficiently.

```
Multiplying Operators:   *, /, mod, rem
Adding Operators:        +, -
Relational Operators:    =, /=, <, ,=, >, >=
Logical Operators:       not, and, or, nand, nor, xor, xnor
```

Continuing the example of the software radio, it was noted that for large amounts of computation, a multiprocessor may be required. In the case that this was required, it would be advantageous to use as many of the same kind of processor as possible. This is illustrated