# DAT093
# Lab Feedback and Guidance 2

**Mehrzad Nejat**

Department of Computer Science & Engineering
Chalmers University of Technology

Sep. 2017

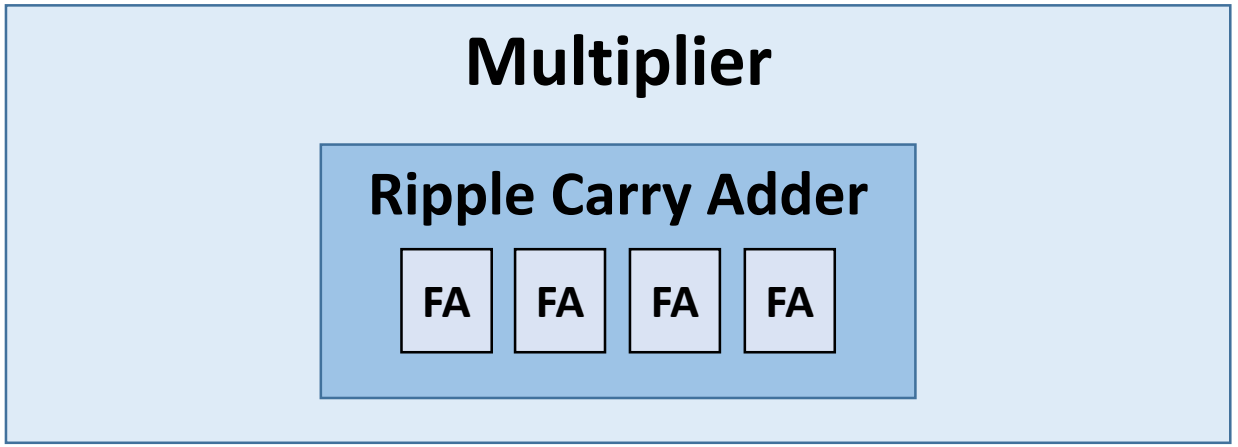
# How to design a complex system in VHDL?

The answer is ORGANIZATION

- Code → Indentation, spacing, …
- Components and hierarchy → Hierarchical Design
- Your mind! → Solve one problem at a time

# Solve one problem at a time

You have been doing that in your hierarchical design

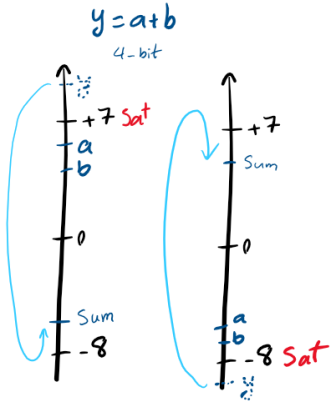


# Solve one problem at a time

You can use the same approach in one design

***Example: Saturation logic***

Step1: Pen and paper




Step2: Decompose the problem
(The question mark technique !)

```
Max_pos<= ?
Max_neg<= ?

y<= Max_pos when ?  else
    Max_neg when ?  else
    sum;
```

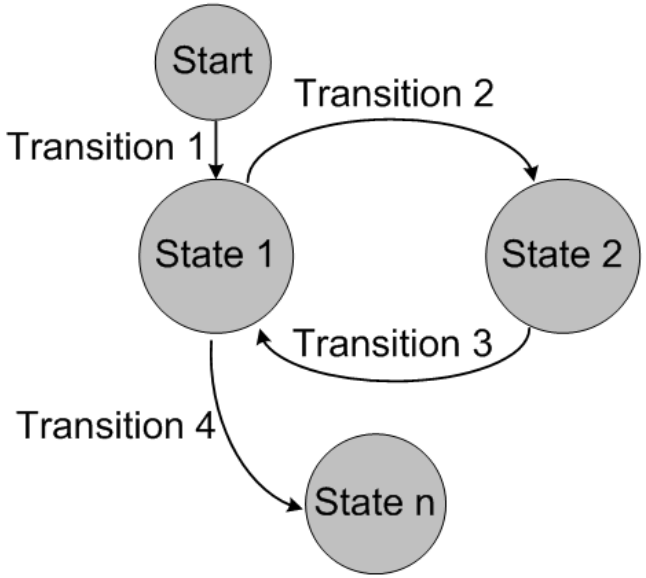
Step3: Solve each sub-problem in English

```
Max_pos<= ?   -- First bit zero, the rest one
Max_neg<= ?   -- First bit one, the rest zero

y<= Max_pos when ?  else   -- when there is overflow and …
    Max_neg when ?  else   -- when there is overflow and …
    sum;
```

Step4: Replace the question marks with VHDL code

# State machine

You can use a state machine to design your sequential systems




Step1: Define your states

Step2: Define what should happen in each state

Step3: Define the transitions between states

# The process statement

You must be able to easily understand the logical flow in your process statement

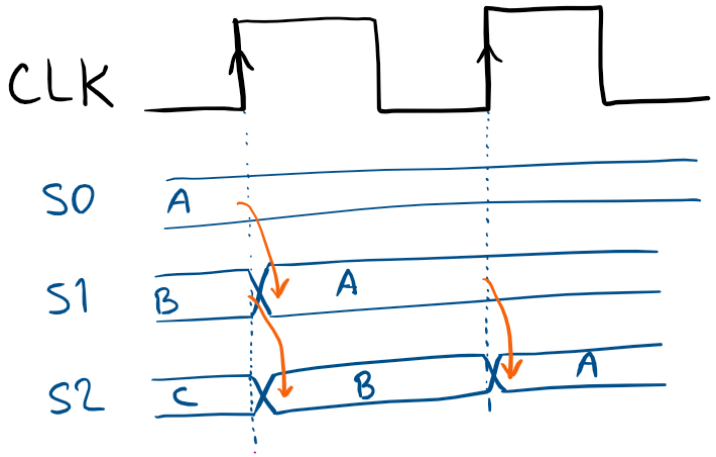
```
Process (clk)
Begin
If rising_edge(clk) then
   if (…) then   -- State1

   elsif (…) then  -- State2

   else   -- State3

   end if;
End if;
```

- Try to use nested “if…elsif…else…end-if” statements rather than consecutive “if…end-if…if…end-if”. You might overwrite something by mistake.
- Assign some value to each process output in any possible condition. Even if you don’t want to change it, assign its previous value
- Always have an “else” condition. Otherwise, you don’t clearly understand what happens if the condition is not true.
- Be careful with the timing of signals!

# The process statement

Be careful with the timing of signals

```
Process (clk)
Begin
If rising_edge(clk) then
   S1<=S0;
   S2<=S1;
End if;
```




Here, at each cycle:
S1 gets the previous value of S0 (the new value of S1),
S2 gets the previous value of S1 (not the new value!)