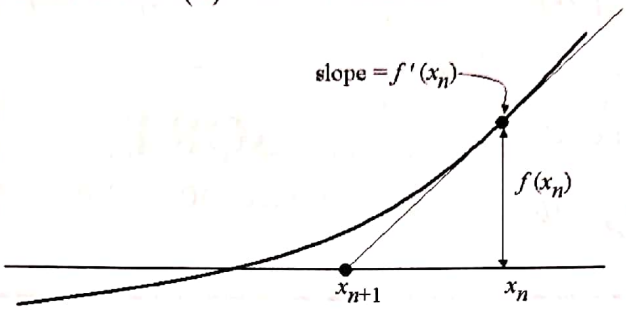
Design of generic square root computing architectures on FPGA using VHDL

The aim of this project is to design on FPGA different architectures implementing two different algorithms for square root computation with different design constraints. These architectures will be designed and tested using VHDL. They will compute the square root of an unsigned input N whose size is 2n bits.
The first considered algorithm is based on the Newton method, whose principle is shown on Figure 1, to compute the root of the $f(x) = x^2 - N$ function.



$$x_{n+1} = x_n - \frac{f'(x_n)}{f(x_n)}$$

Figure 1 : Newton's method principle

Starting from an arbitrary initial guess x_0, a new result closer to the solution is obtained at each iteration using Newton's formula. The algorithm stops when two consecutive iterations give the same result, which is the integer square root of N. Few iterations are required to get the result.

The second algorithm is an iterative algorithm computing sequentially the bits of the result, starting from the MSB. It is detailed on Figure 2.

```
Load N ;
V = 2^(2n-2) ;
Z = 0 ;
For i = n-1 downto 0 do
        Z = Z+V ;
        If (N-Z)>=0 then
                N=N-Z ;
                Z = Z+V ;
        Else
                Z = Z-V ;
        End If ;
        Z = Z/2 ;
        V = V/4 ;
End For ;
Result = Z;
```

Figure 2 : Considered Square root iterative algorithm

The circuit must be useable as a coprocessor by an associated microprocessor. If the designed circuit is sequential, in addition to the *X* input, it shall then feature a *start* input that launches

the computations when its value is 1. It will remain at 1 during the whole computation. Once the computation is over, a *finished* output must be set to 1 to indicate to the microprocessor that it can get the correct result. Once the microprocessor has read the result, the *start* signal is reset and a new X value may be sent for a new computation. Figure 3 summarizes the top-level view of the sqrt circuit.

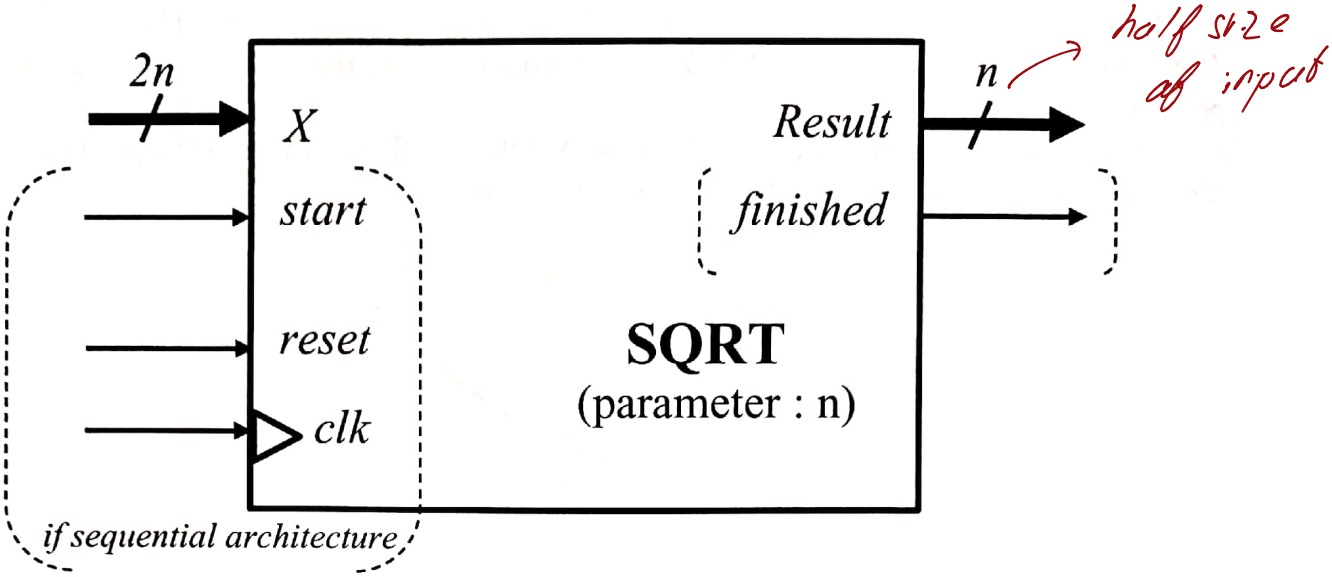


Figure 3 : Top level view of the sqrt circuit

You will have to design several architectures for this circuit and compare their performance in terms of computation duration and circuit size in the case n=32.
In order to do so, you will have, **for each architecture**, to determine the maximum working frequency, the worst-case required number of clock cycles to perform the computations and the resources used in the FPGA.
Of course, each architecture will have to be thoroughly tested beforehand, using testbenches written in VHDL.
The project report must feature the simulation waveforms validating the working of each architecture together with the performance comparison.

1) Behavioral approach :

In this part, all the architectures will be designed in a behavioral way in VHDL **based on only one process.**

Architecture 1 : this architecture is sequential and aims at implementing the algorithm based on Newton's method by minimizing the number of states necessary to perform the computation. Each necessary iteration should be performed in one clock cycle.

Architecture 2 : this architecture is sequential and aims at implementing the second algorithm trying to minimize the number of clock cycles required to perform the computation. Rewrite the algorithm and implement the architecture so that each iteration of the for loop is performed in one clock cycle.

Architecture 3 : Based on architecture 2, implement a combinatorial architecture for sqrt.

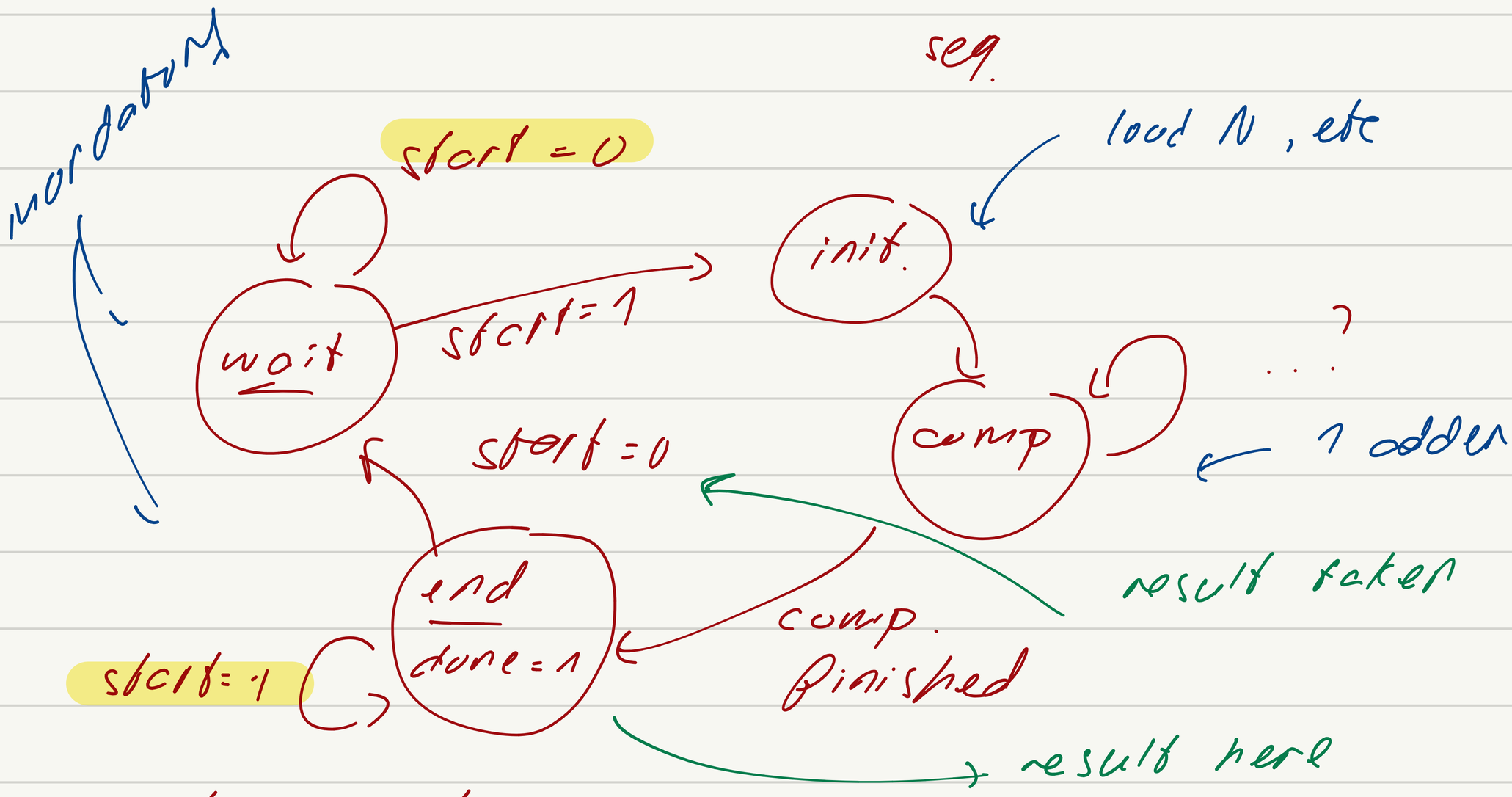
Architecture 4 : Modify architecture 3 to get a pipelined architecture.

2) Structural approach :

The circuits obtained using high-level approach are dependent on the synthesis tools used. If you want to better control the architecture of the obtained circuit, it may be necessary to create a more detailed description of the circuit by explicitly instantiating the desired blocks. This approach will be used to implement the last architecture.

Architecture 5 : This architecture is sequential, based on the second algorithm, and composed of a datapath featuring elementary blocks (like registers, counters, multiplexers …) but only one arithmetic block (an adder/subtracter) to limit the circuit size. Associated to this datapath is a control unit (Finite State Machine) configuring the datapath properly at each computation step. Design all the required elementary blocks (if time is too short, some could be borrowed from your teacher's catalog) and the corresponding control unit and instantiate all these blocks to obtain the desired architecture.

1. seq.
2. combinational
3. pipeline
4. structural → site (limitation 1 adder)

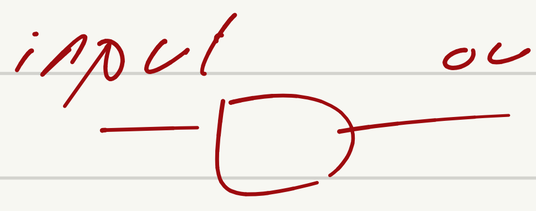


pipeline arch:

- always on → need input (start)

combinatorial arch:

- put registers of input and output (through critical path)
- no sequential



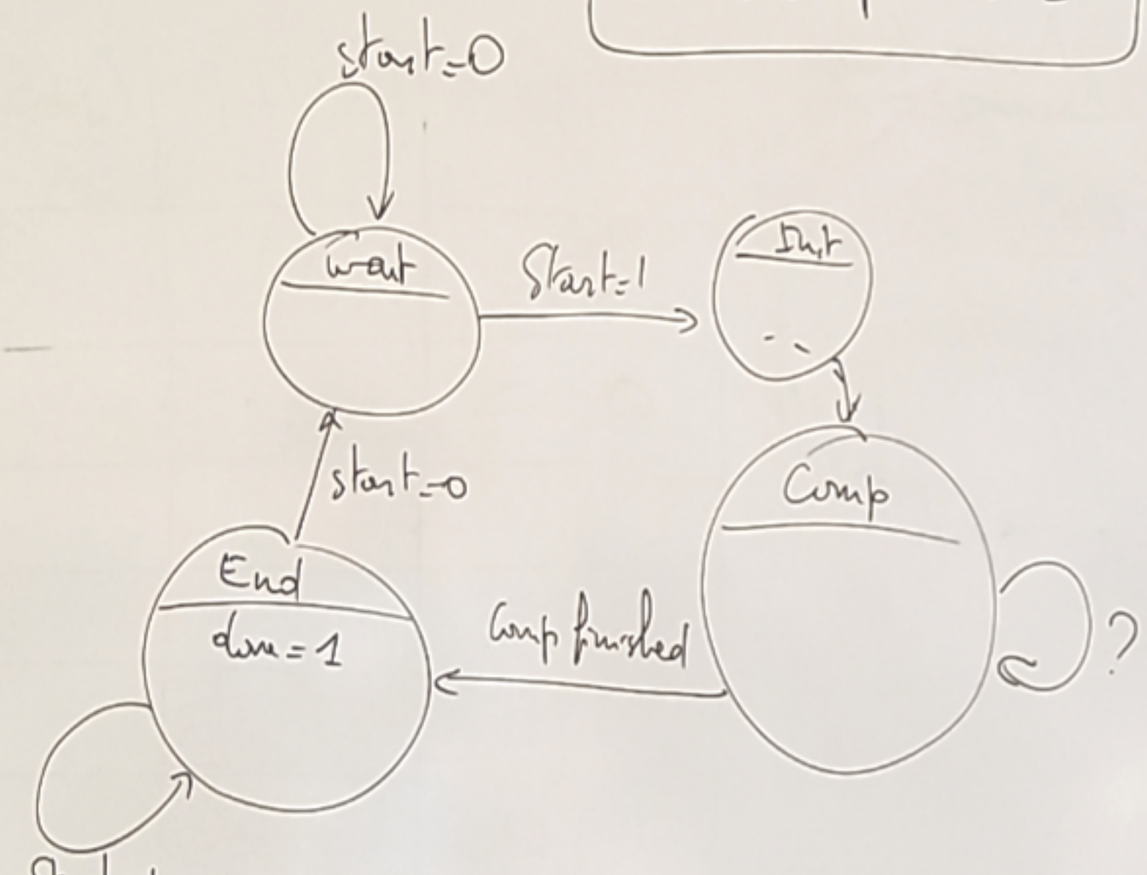
> N (32 bit) generic parameter

of 1 port: 16 bits → duplicate, concatenate

report:

simulation → chains of computation works

1800@tp-de3a1



Quartus Web Edition → 13.0sp1
→ Cyclone 2

login@u-psud.fr

Iterative algorithm

load N
V = 2^{2n-2}
z = 0
for i = n-1 downto 0 loop
  z = z + V
  if N - z ≥ 0 then
    N = N - z
    z = z + V
  else
    z = z - V
  end if
  z = z/2
  V = V/4
end loop
Res = z

N = 4

4 → 0100

V = 4;
z = 0;
then 1 → z = 4;
N - z ≥ 0 → N = 4 - 4 = 0
z = 8
z → 4
V → 1
then 2 → z = 5
N - z < 0 → z - 4
V = 0
z = 2