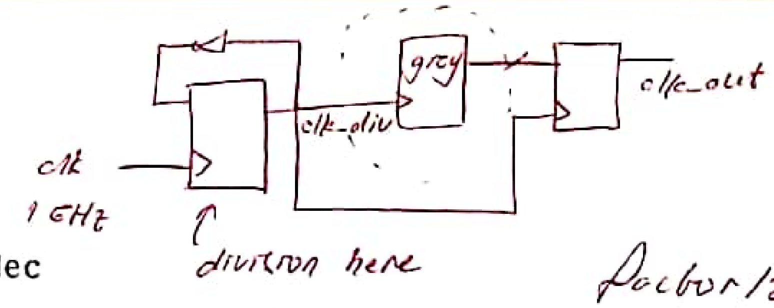


CentraleSupélec

factor/2

clock_div → grey enter

1 ns

2 ns

ICS M2 Degree - 2021-2022 - Advanced Digital Electronics

Practical Session N°9

Synthesis of a frequency divider based on a gray code counter

We now wish to carry out the synthesis of the gray code counter. To do this, we first make the following assumptions:
The factor input will be assumed to be constant or at least not to change in normal operation mode.
We don't care how long it takes to provide the output of the counter.
We wish to operate the counter at a frequency close to 1 Ghz

./GENUS_FULL &

1°) Realize the logical synthesis of the module compteur_gray architecture a4 using the scripts provided by the teacher and available in the GENUS directory.
Once the synthesis is done, open the timing file timing1.txt obtained.
Analyze the result. Do we hold the timings? not really
Note the slack obtained, i.e. the time still to be saved.

2°) We are going to change the parameters of the synthesis tool settings
To do this, edit the file synth_full.tcl and increase the synthesis efforts

del txt check exit

```
set_db syn_global_effort high
set_db syn_opt_effort high
```

relaunch

Restart the synthesis and note the gain obtained

→ 3°) We now change the library of logic gates. We replace the 1.8V gates by 3.3V gates.
We go to the mmmc.tcl file and replace the libraries used :

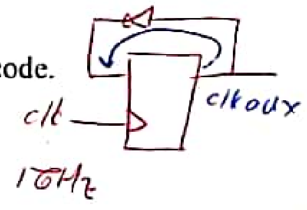
```
create_library_set -name slow_liberty -timing $LIBSET_SLOWHV
create_library_set -name fast_liberty -timing $LIBSET_FASTHV
create_library_set -name typ_liberty -timing  $LIBSET_TYPHV
```

We can notice that the variables $libset_xxx are defined in the file LIB/libset.tcl

Restart the synthesis and note the gain obtained

→ 4°) As we are still far from our objectives we decide to divide the working clock by 2. We introduce a clock divider by 2 in the a5 architecture to get the a5 architecture.
Resimulate with the new architecture and check its correct functioning, then synthesize the result.
Note as always the slack obtained

5°) As we still don't have the constraints, we have to enter and optimize the code.



modify global arch

↳ critical part of min. delay ✓

We are going to introduce in the loop 2 registers cp1 and cp2 which will detect 1 clock cycle in advance the comparison with the factor input.

```
signal cp1,cp2 : std_logic ;

puis dans le processus p1

  cp1 <= '0';      cp2 <= '0';
  if (count(count'left downto 1) = bin2gray(factor-1)) then
    cp1 <= '1';
  end if ;
  if (count(count'left downto 1) = bin2gray(factor/2-1)) then
    cp2 <= '1';
  end if ;
```

Create an a6 architecture.
Introduce into the P1 process the code provided previously, then replace the tests that involved count and factor by the registers cp1 and cp2.
Simulate the a6 architecture, then synthesize it.
Note the significant gain obtained on slack

6°) Conclude on the maximum frequency at which our frequency divider counter will be able to operate.