***Thoughts scribbledoc components***

**CLK\_DIV**

The mclock needs to be halved to make the CLK output which feeds the rest of the circuit. This is the “first” step in the implementation of the entire circuit. An internal variable is initialised with a value of 0, and is inverted every rising edge of the MCLK, and is set as the output of CLK. This leads to the CLK taking two full MCLK cycles to go from 0 to 1, and two more to go back from 1 to 0, etc.

**SYNC**

For the sync circuit, the input signal needs to be fed to the output on the rising edge of the clock signal. This has the effect of synchronising the input to the clock. To do this, two D-flip flops are used. In implementation, this only needs to be two lines of vhdl code and one internal signal, as it achieves the same effect.

First, within a process sensitive to both the CLK and RESET, the reset signal is accounted for, with an if statement. It checks for the reset input bit to be ‘1’ and if so, will set the internal signal and output to ‘0’. Next, if the reset bit is ‘0’ (like most of the time) then the internal signal bit is passed to the output, and the input’s bit is then passed to the internal variable. A signal Is used as it has no delay unlike a signal.

An issue originally encountered, is the reset functionality. The reset was only being applied on the rising edge of the clock cycle. I later discovered it was due the process only being sensitive to the CLK and not the RES signal.

Most of the issues encountered with this circuit lies with the testbench. For low CLK : input frequencies, the output will be synchronised to the CLK, but could be cut-off early leading to the duration of the output to vary. This does seem to be by design, but I will confirm with lecturers next fri.

**FREQUENCY\_CALCULATOR**

DIV50K

First, the div50k needed to be adjusted from the div5k pre existing resource. the Load output was renamed to LD for continuity/consistency, the initial output value was initialised as 0 instead of undefined, and the counter was changed to 50k from 5k. the testbench reflects these changes when simulation is ran.

DIV10

The DIV10 and its testbench components also need some changes:

Ieee numeric std package was not included in the div10 TB, leading to the unsigned initialisation of the signal Q to be flagged as not declared. The DIV10 component definition also was flagged, as there was a duplicate “end DIV10” below “end component” likely from copy-pasting. These were resolved and edits to DIV10 began.

First step is to add CLK as an input and change the sensitivity of the process to CLK and RES from CE and RES, so it only triggers on clock cycles. This way, if on a given rising edge of CLK there is a rising edge of the CE input, then the internal counter gets incremented. The rest of the functionality remains identical.

The TB also needed to be modified. As the input CE will always be synchronised to a CLK cycle, the TB needs to reflect this without having a synchronised input signal. A period of 100ns for CE and 10ns for CLK was chosen, with CLK being initialised as a logic high, meaning every CE rising edge, there will be a CLK rising edge. A correct expected output is then seen where the signal Q increments from 0 to 9, and then outputs a logic high overflow, which will be the CE input of the next DIV10. Q is also reset to 0 for the next cycle to begin.

REGI

As the register component is mostly identical to the existing register component, there doesn’t need to be much in the way of changes except for 2 more inputs and outputs being added, and useless features such as the fre\_out output.