**QPSK simple De-Mapper integrated with fifo:**

In this Exercise I designed a module that receives an input stream of QPSK Symbols, de-map them and deliver an output stream of Bytes representing ASCII characters composing a readable text message.

“Coding Verilog is Fun”

If you have no QPSK digital communication background, don’t worry, all you need to know is that some Analog interface will transfer the radio signal into a stream of symbols, where a QPSK symbol is a Cartesian representation of I and Q which are the X and Y axis respectively. In our implementation I, Q are each a signed value of 8 bits.

Each Symbol should be de-mapped into 2 bits according to this scheme:

תמונה שמכילה עיגול, קו, תרשים, עיצוב

התיאור נוצר באופן אוטומטי

Where I,Q each range from -128 to +127 , and the actual symbol points are at I= +/- 64 , Q= +/- 64.

Starting from reset - upon each received 4 symbols the module should compose and deliver an 8-bit data byte, bit direction is from ‘lsb’ to ‘msb’ meaning the first of each four symbols de-mapped bits go into the least significant 2 bits of the byte and so on.

The data coming out of the QPSK is internally buffered through the fifo.

Initially The data is **coming on an average every 3 cycle** and **fetched out on average every 2 cycles**, so if the fifo is deep enough It is unlikely to become full which will not assert an overflow error message.

**תמונה שמכילה טקסט, צילום מסך, גופן, תרשים

התיאור נוצר באופן אוטומטי**

Code construction order:

1. build each inside module (qpsk, fifo).

2. for each inside module build a testbench with a simple verification environment inside to check that the design work well.

3. verify the testbench correctness by creating a wave form with the inputs, outputs and inside signals in the module.

4. build top module and repeat 2-3.

Attached to this file:

1. All the modules+testbenches+waveforms in separate folders (the full design module+testbench+waveform is inside the project directory).

2. A picture of the waveform of the full\_design testbench to show correctness of the code (the signals in the picture are separated by dividers to distinguish between the signals that relevant to each internal module)

3. A picture of the transcript with the displayed message, to show correctness of the code.

The code was written in Verilog language in the Modelsim simulator.