ON THE USE OF RAPID PROTOTYPING FOR DESIGNING PCM/FM DEMODULATORS IN FPGAS

Michael Rice, Brent Nelson, Marc Padilla, Jared Havican
Department of Electrical & Computer Engineering
Brigham Young University, Provo, Utah, USA

ABSTRACT

This paper describes the use of an efficient FPGA design flow, called *Ogre*, developed at BYU to design and implement PCM/FM demodulators. *Ogre* exploits the notion of reuse by taking advantage of a library of specially designed cores parameterized by XML metadata. A judicious choice of library cores, targeted to signal processing functions common to sampled data modulators and demodulators, reduces the design and test cycle time. We demonstrate this by using the tool to construct rapid prototypes of three different versions of FM demodulators and show that the bit error rate performance is comparable to demodulators on the market today.

KEY WORDS

FPGA, PCM/FM, Demodulators, Bit Synchronization

INTRODUCTION

The signal processing functions associated with modulation and demodulation continue to migrate from the continuous-time domain to the discrete-time domain. This trend is due to a number of factors that exploit the advantages of sampled-data systems. Sampled data systems are far more flexible than their continuous-time counterparts, especially in the functionality available to the system designer. Sampled data systems also offer the prospect of *reconfigurability* which enables highly efficient multimode operation for both modulators and demodulators. The key factors that have enabled this trend are the availability of high rate, relatively high precision, analog-to-digital converters (ADCs) and high-performance discrete-time processors such as programmable DSPs, FPGAs, and ASICs.

The programmable DSP is the most flexible and the easiest to program, but achieves these advantages at the cost of performance. The ASIC offers the best performance, but is not flexible (at best, ASICs can provide limited parameterization) and the design process is very long and costly. FPGAs offer an attractive middle ground: the FPGAs offer flexibility and "reprogrammability" like the programmable DSP, and have performance that can rival that of the ASIC.

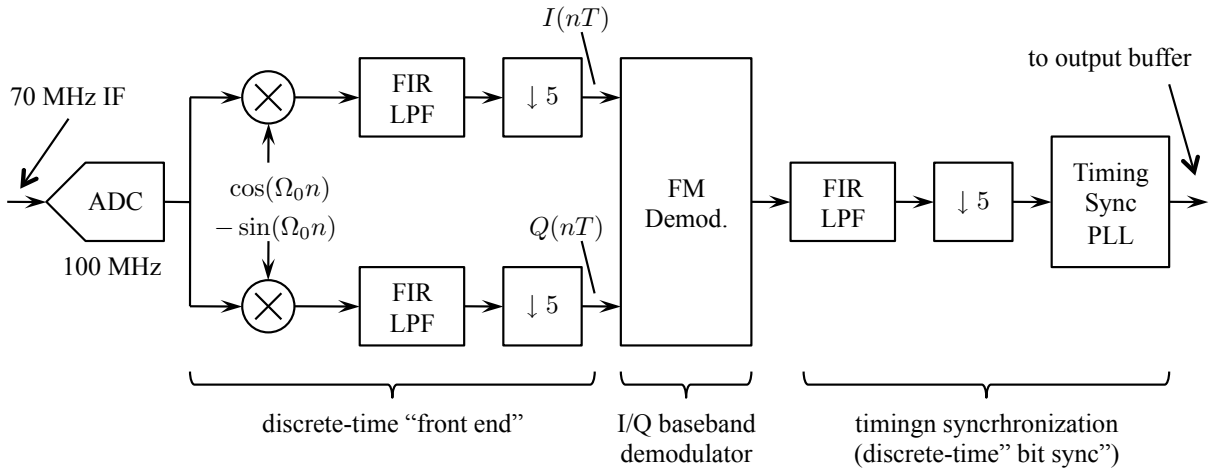


Figure 1: PCM/FM demodulator block diagram, divided into sections based on clock domains.

However, there are challenges in developing communication systems with FPGAs. Design and test processes can be lengthy and tedious. Design productivity can be increased through "reuse" [1]. Reuse (or "using again") leverages previous FPGA designs in much the same way a computer programmer leverages subroutines available in a code library. The key to reuse is a library of cores that offer sufficient flexibility to be useful and are not too small or too large.

To address this issue the Configurable Computing Laboratory at Brigham Young University has developed a tool that enables rapid prototyping of radios. The tool, called *Ogre*, is a design flow for FPGAs that exploits a library of parameterizable cores specifically designed for the tasks needed by modulators and demodulators. Each core is also accompanied by XML metadata providing a description of the core. The *Ogre* tool flow leverages this metadata to intelligently interface cores which have been connected in a design.

This paper reports on the application of the *Ogre* tool to design PCM/FM demodulators. The design flow and testing are described in the following sections. Laboratory experiments show that the *Ogre* design flow is able to produce good PCM/FM demodulators in a matter of hours without sacrificing performance.

BASIC PCM/FM DESIGN

The basic outline of the PCM/FM demodulator is illustrated in Figure 1. The 1 Mbit/s PCM/FM signal at an IF of 70 MHz is presented to an ADC sampling at 100 Msamples/s. The sampled signal is translated to I/Q baseband using a discrete-time quadrature mixer. The I/Q baseband samples are downsampled to 20 Msamples/s and presented to the FM demodulator. The FM demodulator output is downsampled to 4 Msamples/s to produce a PCM pulse train at 4 samples/bit. Timing synchronization is performed by a timing synchronization PLL. The target platform for this design is the Nallatech XTremeDSP board illustrated in Figure 2.

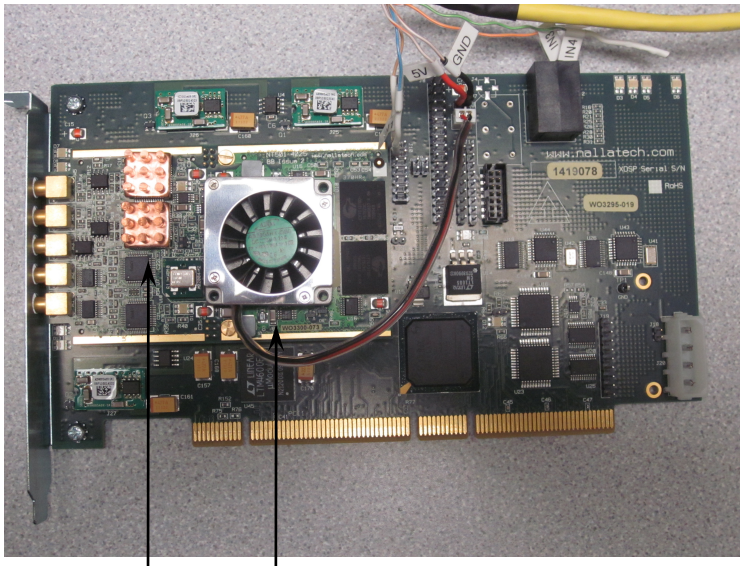


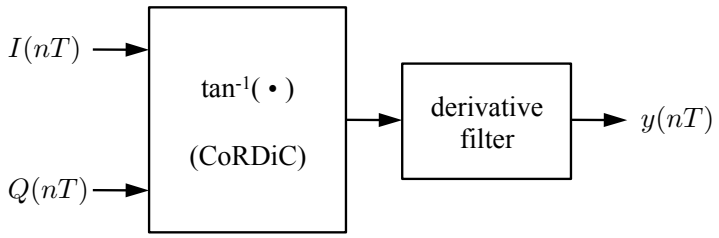
Figure 2: The target platform for the PCM/FM demodulator: the Xilinx/Nallatech XtremeDSP board.

Three options for the FM demodulator were explored. Block diagrams of these options are shown in Figure 3. We showed in [2] that the three options have the same SNR performance and provide an area/clock-rate tradeoff. The options shown in Figure 3 (a) and (b) compute the derivative of the instantaneous phase of the input I/Q sample pairs and represent the discrete-time equivalent of the limiter discriminator demodulator. These circuits achieve the highest clock rates, but also require the most area [2]. The option shown in Figure 3 (c) is a discrete-time phase lock loop and requires the smallest circuit area but cannot operate at as high a clock rate as the other two options [2].

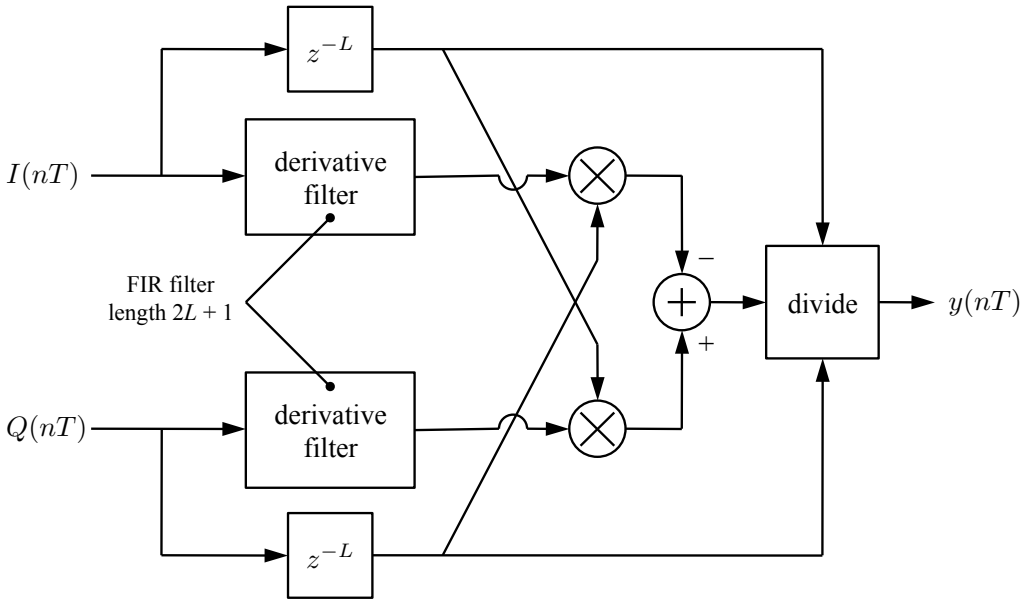
The timing PLL is shown in Figure 4. It is a traditional discrete-time timing synchronizer using the early-late timing error detector [3] operating at 4 samples/bit.

DESIGNING WITH THE OGRE TOOL

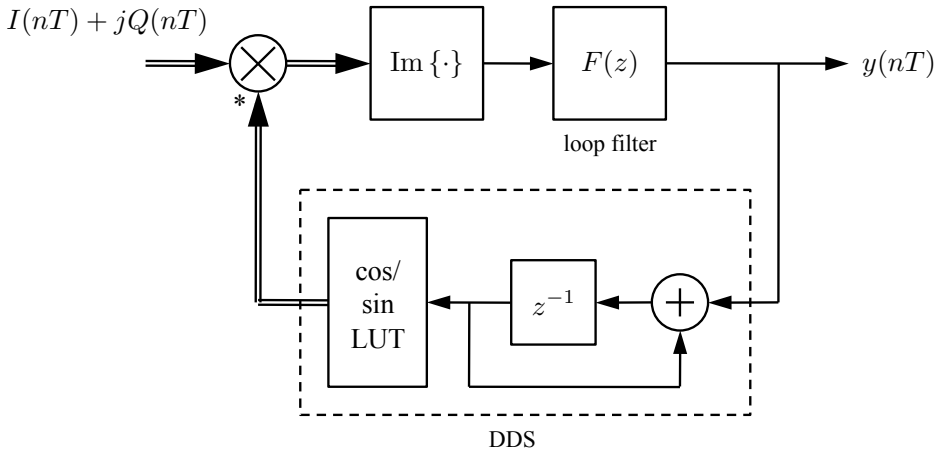
The PLL option of the FM demodulator was designed using the *Ogre* design tool. Since both options (a) and (b) are made up of commonly used blocks – FIR filters, CoRDiC, divider, multlipliers – these designs were assembled using cores from Xilinx CORE Generator. Xilinx CORE Generator provides well-parameterized blocks for systems such as these. However, the option (c) requires much different blocks, such as a direct digital synthesizer (DDS) and loop filter. In this case, the *Ogre* tool was very useful in providing an environment in which blocks of this type could be found and connected. A library of blocks had already been created specifically targeting PLL-style radio systems (see [1]). A few blocks from this library were used to create the final option (c) fm demodulator shown in Figure 1.



(a)



(b)



(c)

Figure 3: The option for the FM demodulator.

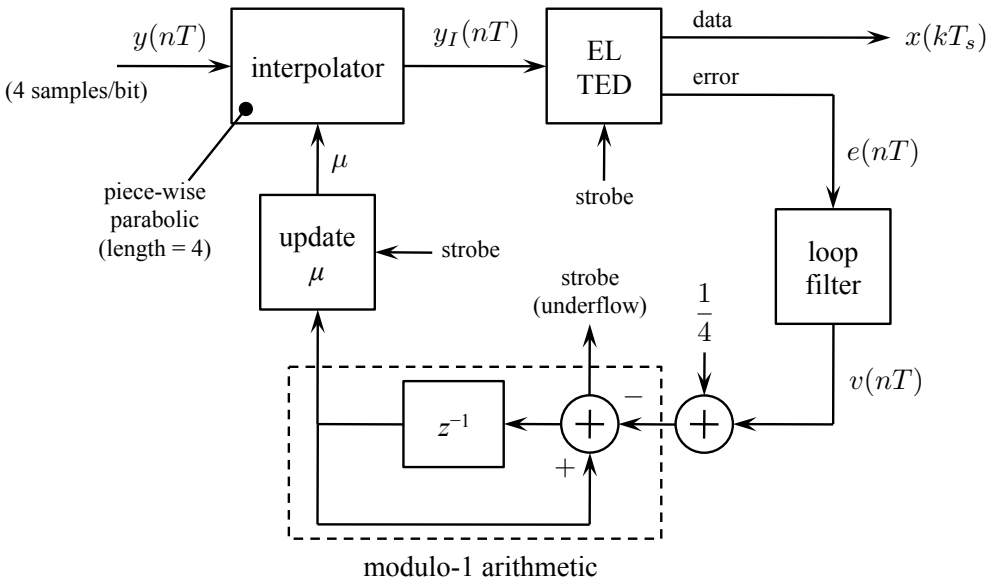


Figure 4: Detailed block diagram of the bit timing synchronization PLL.

The *Ogre* tool offers many benefits to the designer, some of which were used in the creation of this design. One nice feature is that of connecting signals such as the clock input. This signal, which is required by most blocks, does not have to be hooked up manually in the *Ogre* design. It is simply left disconnected in the design. During the VHDL generation, all clock inputs are merged into one top-level input. This is also true of the clock enable and reset signals. A screen shot of the Ogre design environment for the discrete-time PLL-based FM demodulator of Figure 3 (c) is illustrated in Figure 5.

Along with merging common inputs, such as the clock, the *Ogre* tool generates a state machine to enable every block at the correct time. It does this by creating a schedule, based on information from the XML metadata, of when every block requires its inputs and when each block's outputs are ready. Once this schedule is created, VHDL is generated to enable the validIn port for each block at the appropriate time. This feature is and has been especially useful for pipelining designs which need to run at higher clock rates. The designer may simply add registers anywhere in the design for timing closure to be met. These registers are taken into account by the *Ogre* tool when the schedule is created so that the data in the design still flows appropriately. In this way, the functionality of the original design is maintained while allowing the design to be clocked at much higher rates. In the PLL option design, this feature was not necessary due to the simplicity of the schedule. The *Ogre* tool was able to figure this out and a state machine was generated which enabled every block on every cycle.

The parameters on each block in the *Ogre* tool are easily updated. Figure 6 shows an example of how parameters are set for a certain block, in this case the loop filter block. Once a block in the design is clicked, the current parameters for the block are shown and can be changed by the user.

The information regarding what parameters exist and what values are valid for each parameter is found in the XML metadata accompanying each block. Parameters range from low-level things such as bit widths, to higher-level properties such as loop bandwidth. With these highly

parameterized blocks and the ease of changing parameters, blocks become very reusable to designers.

Overall, the design process for the PLL option was very much simplified by the use of *Ogre*. With the ability to reuse blocks and with *Ogre* doing much of the work itself, the design completed in less than an hour. Of course, this did not represent the complete design. The generated VHDL had to be integrated with the rest of the system for the FPGA to be correctly configured. However, the overall design time was reduced due to the use of *Ogre* on this section of the design.

When the designer is finished connecting the blocks in their design, the "BYU Interface Synthesis" block (which is present in every *Ogre* design) is clicked to reveal the *Ogre* VHDL generation interface. An example is shown in Figure 7. Once the output directory is specified, the designer clicks the "Generate" button to start the process. It is at this point that the design is reviewed, the schedule is created, and the top-level VHDL is output along with VHDL for the state machine. The VHDL for all of the library cores used in the design is also output to provide the designer with everything necessary to use the new design.

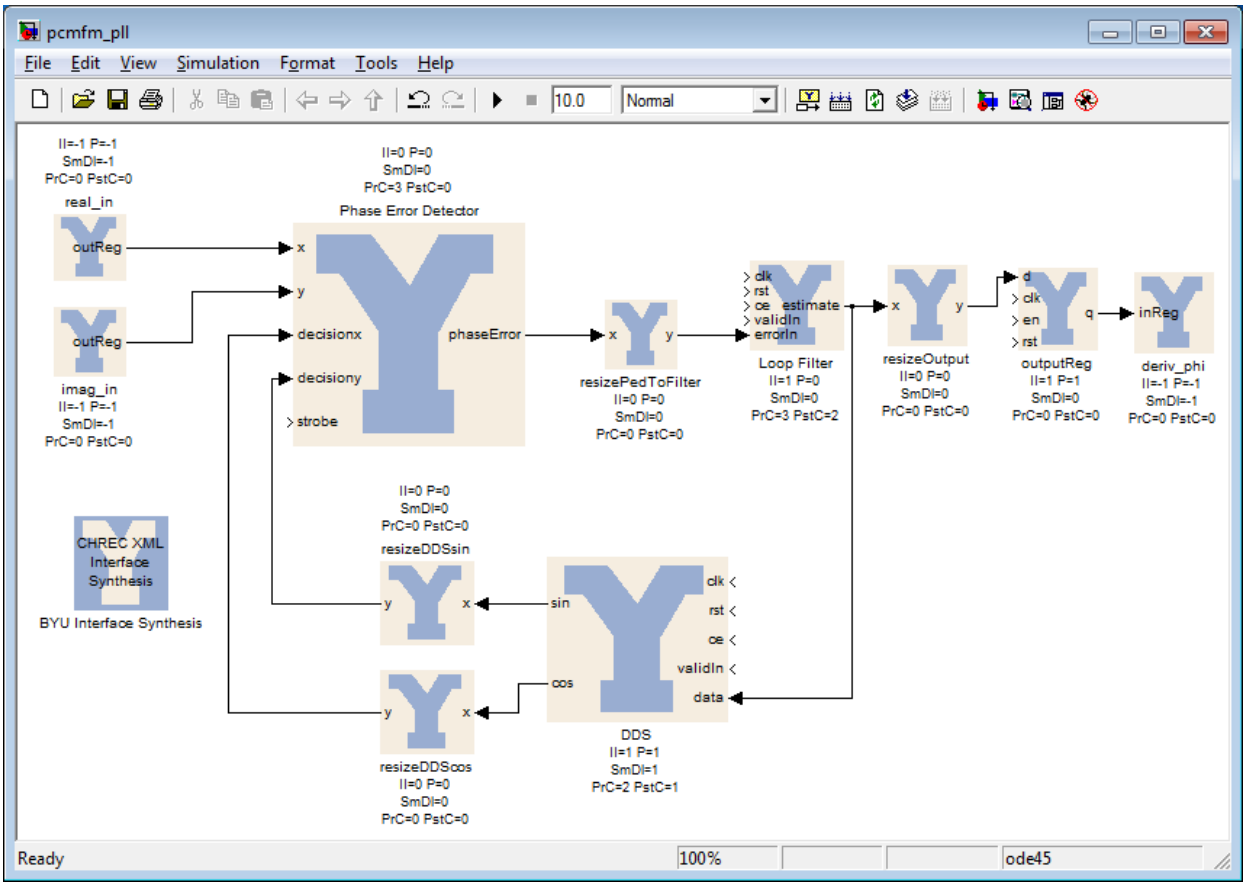


Figure 5: The discrete-time PLL-based FM demodulator of Figure 3.(c) using the *Ogre* tool. The VHDL code generated from this model was used in the final design.

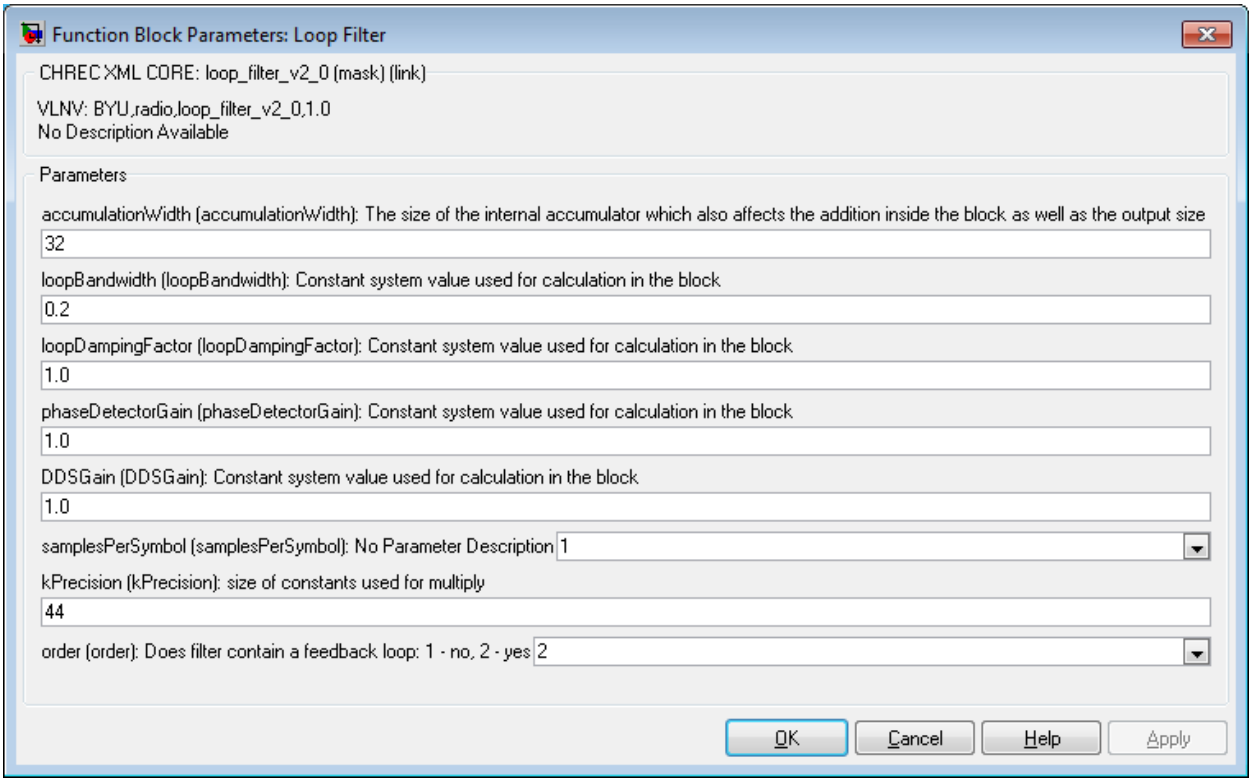


Figure 6: Parameter window for Loop Filter (loop_filter_v2_0) library block.

LABORATORY TEST RESTLTS

Each receiver design was tested in hardware using the setup shown in Figure 8. A tri-mode telemetry transmitter from Quasonix was used as the PCM/FM source. The data source was set to the internally generated length-(2^{15}-1) PN sequence and the carrier frequency was set to 2255 MHz. The resulting signal was mixed to 70 MHz using an LO and mixer as shown. A calibrated noise source was used to set the desired E_b/N_0. A modest LNA was used to set the signal level as required by the ADC. The ADC and FPGA were housed on the Nallatech/Xilinx XtremeDSP board. The ADC operated at 100 Msamples/s and the FPGA was a Virtex4 (XC4VSX35-10FF668). Data and clock were output from the FPGA board and used by the bit error rate test set to measure the bit error rate performance. A photograph of the experiment is shown in Figure 9.

All three designs used a pair of identical low-pass FIR filters as shown in Figure 1. This filter plays the role of the IF filter in more traditional analog designs and controls the trade-off between intersymbol interference and noise power [4]. The low-pass filter was a length-469 FIR filter with an equivalent 3-dB bandwidth of 200 kHz and a transition bandwidth of 678 kHz. The PLL-based demodulator was a second order loop with a damping constant of 1 and closed loop bandwidth of 200 kHz.

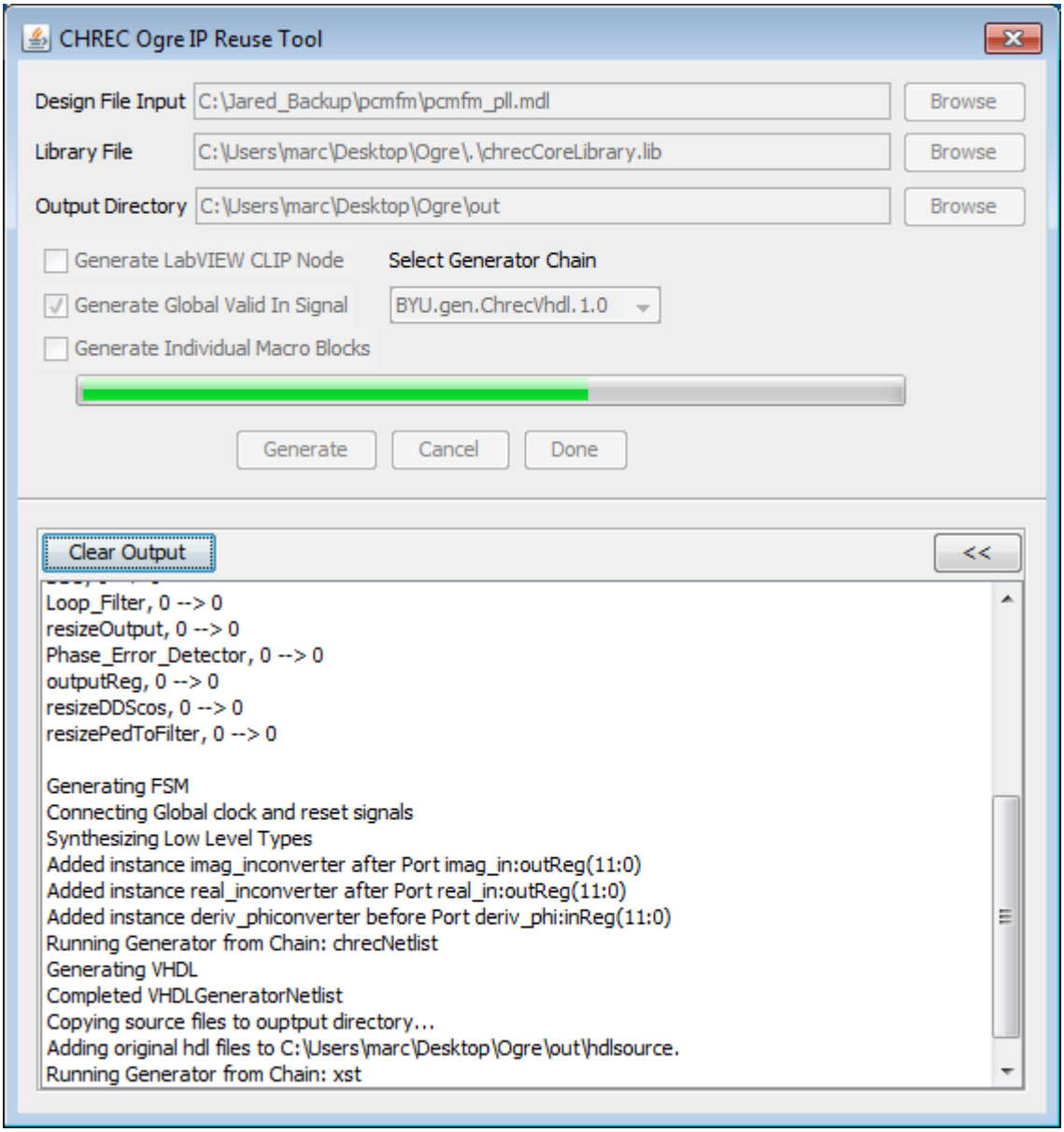


Figure 7: Ogre tool VHDL generation.

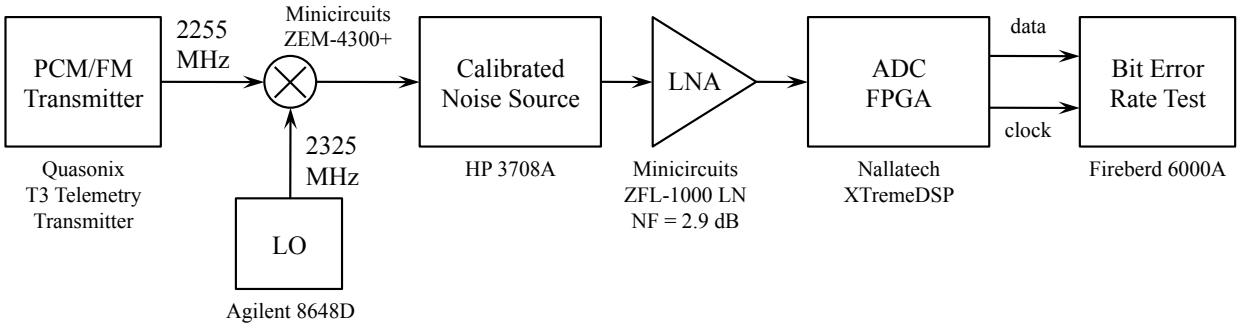


Figure 8: Laboratory test configuration.

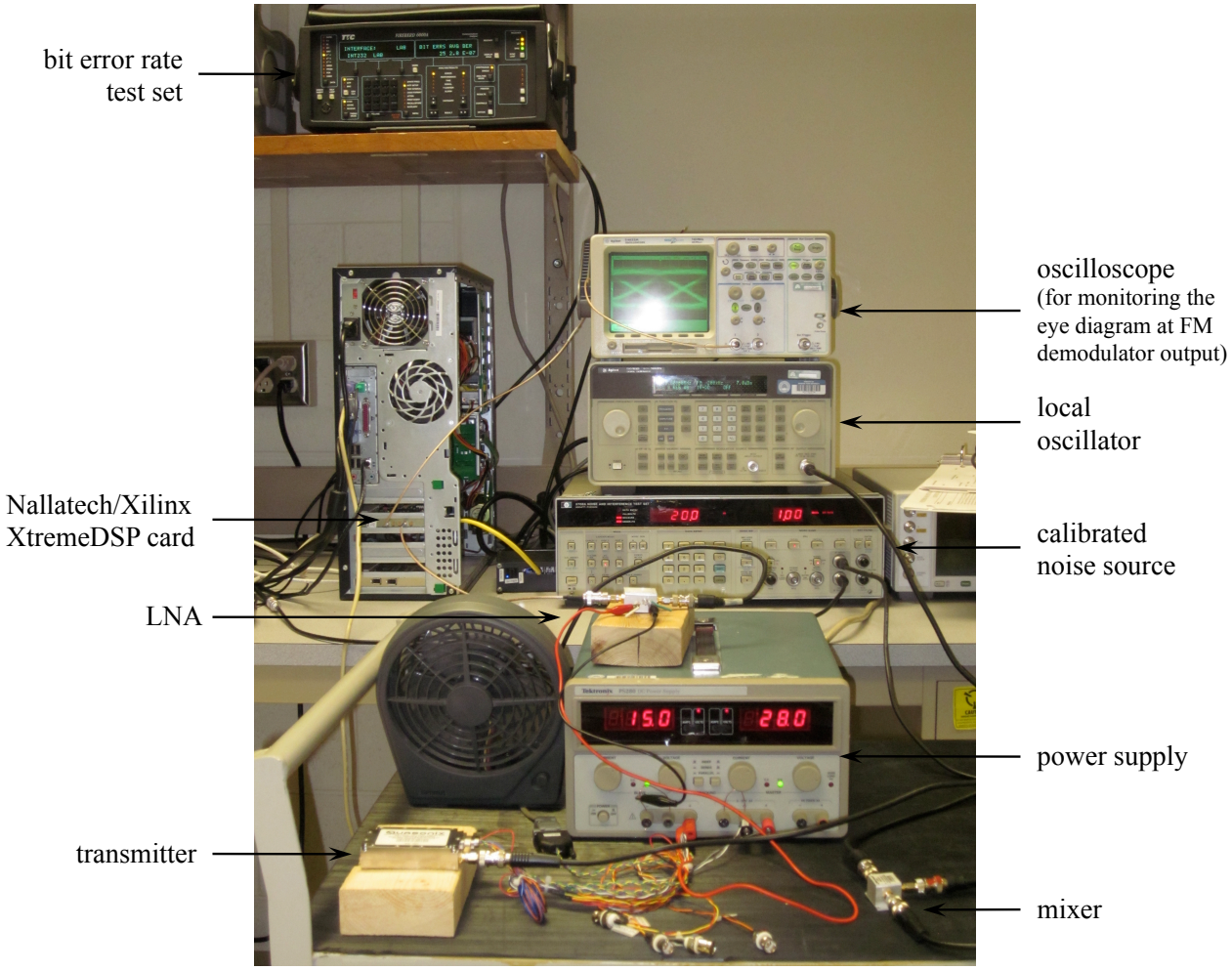


Figure 9: A photograph of the laboratory test configuration for the PCM/FM demodulators.

The test results are summarized by the plots in Figure 10. The only difference between the three approaches is the FM demodulator. The tests results show that the discrete-time versions of the limiter-discriminator shown in Figure 3 (a) and (b) produce essentially the same bit error rate performance. The PLL-based demodulator has a slightly higher bit error rate: about 0.6 dB inferior to the limiter-discriminator approaches. A reference curve is also included in Figure 10. The reference curve is derived from Figure 2-10 of the 199-06 Telemetry Applications Handbook [4]. The relationship between the BER performance of the rapid-prototype demodulator and the reference curves shows that the BER performance of the rapid-prototype is comparable to the commercially available demodulator used to generate the data in the Telemetry Applications Handbook.

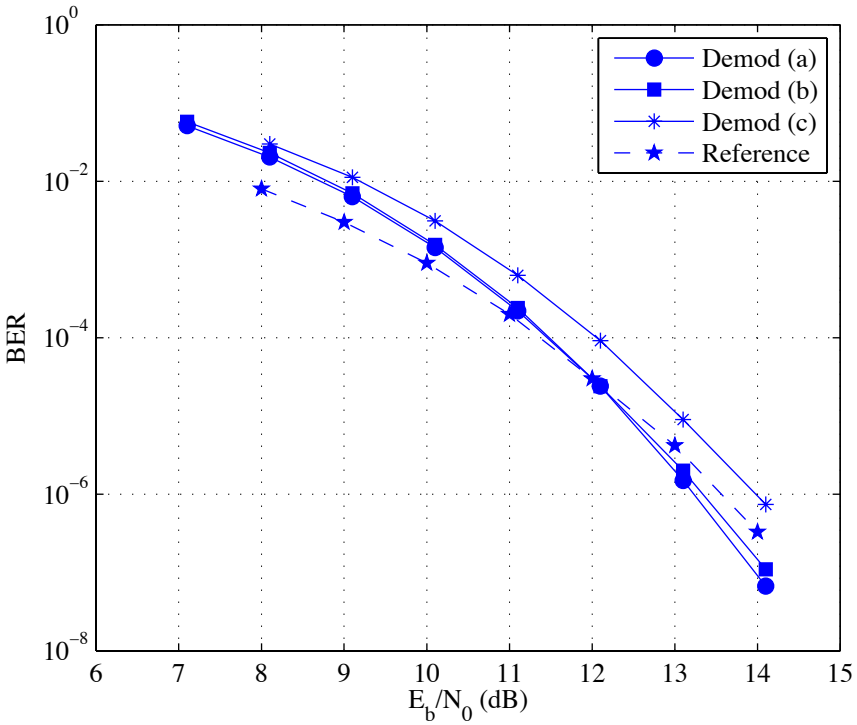


Figure 10: Laboratory test results for the PCM/FM demodulator using the three FM demodulators outlined in Figure 3. The reference curve is from Figure 2-10 of [4].

CONCLUSION

In this paper we described a rapid prototyping environment for FPGAs specifically targeted to modulators and demodulators. This tool, called *Ogre*, reduces the design and test cycle times by exploiting reuse based on a carefully chosen library of cores. We applied the design environment to design three different versions of a PCM/FM demodulator in a few hours. Laboratory tests showed that the bit error rate of the PCM/FM demodulators is comparable to those available on the market today. Thus, the rapid prototyping does not sacrifice performance.

REFERENCES

[1] A. Arnesen, et al, "Increasing Design Productivity Through Core Reuse, Meta-Data Encapsulation, and Synthesis," Proc. of Intl. Conference on Field-Programmable Logic and Applications (FPL), Milano, Italy, Aug. 31 – Sep. 2, 2010.

[2] M. Rice, M. Padilla, and B. Nelson, "On FM Demodulators in Software-Defined Radios Using FPGAs," Proc. of Military Communications Conference (MILCOM), Boston, MA, Oct. 18-21, 2009.

[3] Rice, M. Digital Communications: A Discrete-Time Approach, Prentice-Hall, and Upper Saddle River, NJ, 2009.

[4] Range Commanders Council Telemetry Group; Document 119-06 Telemetry Applications Handbook, 2006.