MULTI-GHZ SOFTWARE AND HARDWARE PLATFORM FOR SOFTWARE DEFINED RADIO

Joseph Rothman (BEEcube Inc, Fremont, CA, USA, joseph@beecube.com)
Chen Chang (BEEcube Inc, Fremont, CA, USA, chen@beecube.com)
Kevin Camera (BEEcube Inc, Fremont, CA, USA, kcamera@beecube.com)

ABSTRACT

Combined with BEEcube's 3rd generation Xilinx Virtex-5 FPGA based hardware platform, the BEE3[1], an integrated set of BEEcube solutions for system implementation enables implementation of a wide range of high-performance real-time military and defense applications, including signal intelligence, signal warfare, software defined radio, MIMO communications, radar, and many more.

The paper presents BEEcube's SDR reference design that highlights BEE3 as an SDR prototyping platform, featuring an FPGA-based continuous wideband vector signal generator, with real-time software control using Wind River's VxWorks [2] over Gigabit Ethernet. Carrier frequency tones ranging from 0 to 2GHz highlight the inherent wideband capability of BEE3's I/Q 2Gsps DAC.

The BEE3 ADC expansion board simultaneously captures the analog output, with data being streamed and displayed directly in the Matlab™ [3] environment via the Ethernet data transfer libraries provided by BEEcube's distributed Nectar OS™ [4] embedded and host-side software solution. BEE3's ADC can sample up to 3GHz, offering true direct RF sampling capability. Through BEE3 Easy Algorithm Deployment coupled with high-speed I/O and infrastructure, the BEE3 system software allows algorithm designers without any RTL or low-level hardware design knowledge to easily program the target BEE3 system.

Advanced signal processing algorithms and complete SDR implementations can be rapidly prototyped on the BEE3 system, running at hundreds-of-MHz clock rates. For deployment, the same design can be easily retargeted in the BEEcube Platform Studio (BPS) [5] design environment to fit into various hardware platforms with different form factors, capabilities, and FPGA technologies.

1. INTRODUCTION

Radio communication remains the simplest and the most flexible method to locally communicate in the battlefield today; however, frequency jamming, channel availability, contention and even spectrum capacity management remain relevant. Fortunately, new real-time and dynamic opportunities for solving these challenges exist in the form of Software Defined Radios (SDR) and Cognitive Radio strategies.

Radio communication, like most electronic technologies today, is leveraging the advantages of moving analog concepts to the digital domain, thus forming a new field of Digital Communication Design. With the move to digital, additional logic can be applied in real-time to both signal transmission as well as the information being carried. Though first generation (e.g. SDR) systems have been successfully deployed, new design ideas are far from exhausted. The challenge is having a vehicle to test new digital radio concepts and algorithms on an efficient prototyping platform that can handle the real-time multi-GHz environment.

The BEECube BEE3 system provides a flexible platform for design, implementation, and validation of next generation wideband SDR systems. Each BEE3 system provides 4 Xilinx Virtex-5 SX95T FPGA chips for DSP processing, two full 3GHz analog inputs, and two 2GHz analog outputs on two independent ADC and DAC modules. These ADC/DAC modules can be used in direct IF/RF synthesis applications. Despite the 3GHz/2GHz sample rate of the ADC/DAC chip, FPGA processing can keep up with the ADC/DAC interfaces by using a built-in multiplexer/demultiplexer feature combined with on-FPGA DDR I/O resources. For example, the 3GHz ADC is connected to each FPGA with 4 parallel DDR750 digital 8-bit channels, and on the FPGA the internal signal path provides the end user with 8 parallel 8-bit channels running at 375MHz each; well within the 450MHz operating frequency of the FPGA device. This allows end users to use the BEE3 system to validate full end-to-end SDR systems running at real-time, in excess of full multi-GHz analog bandwidth.

In addition to the hardware features listed above, the BEE3 unique software design environment—BEEcube Platform Studio (BPS)—is a Mathworks Simulink based turn-key algorithm to real-time implementation tool flow. Users can turn their original SDR communication algorithm simulation models in Simulink into actual hardware implementation, but still retain programmability of software parameters.

To further enhance the programmability and debugging of the system, the embedded component of BEEcube's Nectar OS provides an extensible shell environment where the user can interact with all BPS-generated cores at runtime. Each BPS core provides a driver API which can be used to write custom code to manage and monitor the system under test directly on the FPGA. In addition, Nectar OS can optionally provide a network service for control of all software components from a remote workstation.

In this paper we explore a SDR oriented reference design using the BEE3. The purpose of this design is not to provide a fully functional SDR communication radio, but to demonstrate the various pieces need to construct a multi-GHz wide band SDR radio with real-time implementation on the BEE3-W platform.

In a typical SDR system, the carrier frequency and waveform modulation schemes are the two most important parameters to remain flexible. In the BEE3 SDR reference design presented in this paper, carrier frequency is digitally synthesized on FPGAs based on the Direct Digital Synthesizer (DDS) core from Xilinx.

Unlike traditional low-bandwidth DDS, in a wide-band system, since the ADC/DAC interface has 8 parallel equally phased digital channels in order to produce the full multi-GHz analog coverage, the wide-band DDS also needs to be digitally phased to match the input/output requirements. Instead of hard locking each DDS core's phase increments (controlling the frequency) and relative phase to other DDS cores (controlling the phase alignment), the BEE3 system combined with Nectar OS allows the end user to set these parameters at run-time, hence controlling the exact carrier frequency. The BPS software automatically maps software control registers connected to each of the DDS cores' hardware control ports. A user defined C program was written to provide the end user with command line access to modify carrier frequency with the BEEcube Nectar OS running on each FPGA in the BEE3 system. This program takes the user input of frequency, and then calculates in software the proper phase increments and offset for each of the 8 DDS cores, and then toggles the corresponding DDS control signals to load the parameters to each core. With networking support enabled in Nectar OS, this same functionality is available to a remote control host, for example a management RTOS such as VxWorks.

Modulation schemes can also be modified on the transmitter side by using the BPS shared memory feature. On the output of the each of the eight parallel DDS cores, the carrier signal is digitally modulated with a periodic transmission signal stored in 8 shared memory Block RAM (BRAM) components, one for each DDS core. Users can upload arbitrary waveforms to each of the shared memories and observe the corresponding modulation on the analog output.

On the receiver side, the parallel ADC inputs are directly captured in a similar shared BRAM scheme. Nectar OS fully controls the data capture, which can be triggered externally by the user via the embedded shell or Ethernet. In addition, once a complete window of data has been captured into shared BRAM, the data samples from the ADC can be streamed to a host computer for further analysis at very high speed.

2. SDR EXAMPLE USING BEECUBE'S FPGA PROTOTYPING PLATFORM

In this example, we use the BEE3 development and deployment platform as an SDR prototyping platform. The BEE3 was developed out of research conducted at the Berkeley Wireless Research Center (BWRC) at the University of California, Berkeley, and is a real-time, rack mountable/table top symmetrical multi-FPGA platform. The objective is to demonstrate the use of a platform that allows for flexible algorithm and feature set definitions permitting various and changing mission critical needs.

2.1. The BEE3 Platform and BPS Design Environment

BEEcube's third generation BEE (Berkeley Emulation Engine), the BEE3 [1], is a multiple FPGA platform that is specially suited for prototyping of signal processing algorithms and applications. In this demonstration, we have used a single BEE3 module consisting of four Xilinx Virtex-5 FPGAs. The module has a capacity of 5M ASIC gates. The four FPGAs are interconnected with a ring and each includes two channels of DDR2-400 memory. Each FPGA quadrant has a fully symmetrical design, including identical memory and I/O interfaces. This symmetrical architecture allows both a high level of redundancy for fault-tolerant processing, as well as complete implementation compatibility regardless of which FPGA quadrant is programmed with a particular design. Multiple high-speed data interfaces available on each module include: 160 Gbps SERDES, Quad x8 PCI Express, Quad 1000BASE-T Ethernet, and Quad 40-pair LVDS QSH expansion slots. Each module also has a net capability of 64GB of DRAM.

BEEcube Platform Studio (BPS) [5] is a system-level, hardware/software co-development environment built on top of the MathWorks™ Simulink® framework. BPS provides automatic generation of all platform specific hardware interfaces and corresponding software drivers. Months of engineering tasks to convert complex DSP algorithms to implementation can be achieved through BPS in a matter of days, all without requiring user knowledge of the low level FPGA implementation details, such as high speed I/O interfaces, timing closure, HW/SW interfaces, and IP integration issues.

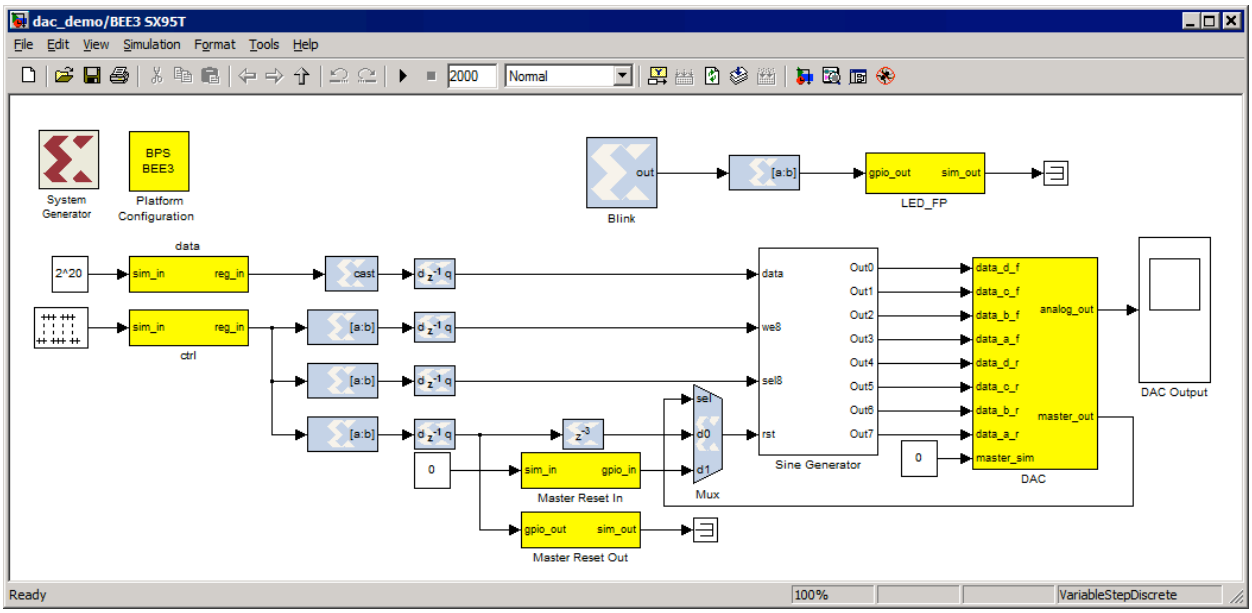

Figure 1: DAC Signal source model built using BEEcube Platform Studio (BPS)

The BEE3 hardware platform also supports ADC and DAC expansion boards. The ADC expansion board is a dual channel 3 GSps model using two National Semiconductor ADC083000 ADCs with independent clock, data, reset and trigger SMA inputs for each ADC. The quad channel 1.5 GSps version uses the ADC08D1500 device. All ADC boards support 8-bit resolution per channel per ADC/FPGA. The analog sample rate supported is 1000 to 3000 MHz, with a balun input bandwidth of 30 to 1800 MHz.

The DAC expansion board features a dual 2 GSps DAC from Teledyne (RDA112M4MSLPD) supporting dual independent clock inputs and data outputs. The resolution is 9-bit up to 2 GSps (with a 4:1 mux) or 12 bit resolution up to 1.5 Gsps (with a 2:1 mux). The design presented in this paper uses the 9-bit, 4:1 multiplexed operating mode of the DAC.

3. THE SDR REFERENCE DESIGN ARCHITECTURE

Using the BEE3 FPGA prototyping platform we are able to support a continuous wideband vector signal generator design and generate sample real-time runs. Controlled by software via a control host (e.g. VxWorks) over standard Gigabit Ethernet communication, the FPGA prototyping system can handle dynamic carrier frequency tone sweeps ranging from 0 to 2 GHz. With 2 Gbps I/O DACs, wideband capability can be handled natively. Coupled with an ADC expansion board, simultaneous capture of the analog output can be directly displayed in real-time using standard low-cost algorithm display tools, such as Matlab. The ADC solution can sample up to 3GHz, offering true direct RF sampling capability.

This SDR reference design uses only 2 of the 4 BEE3 FPGAs, leaving plenty of logic resources for other radio components. The first FPGA (FPGA-A) in our design is used as a 2 GSps DAC signal source. FPGA-A generates the carrier tone, based on a numerically controlled oscillator (NCO), and contains an arbitrary modulation pattern, up to 16,384 samples in length. The modulation pattern is held in FPGA block memory (BRAM) and can be dynamically updated at runtime via a UART or Gigabit Ethernet link from a remote host using integrated software interfaces generated automatically by BPS.

The second FPGA (FPGA-B) is associated with the 3GSps ADC module and functions as a virtual oscilloscope. FPGA-B captures data into its BRAM in 16,834 sample windows. The captured data can be read via a UART or Gigabit Ethernet into Matlab running on the host for direct analysis.

4. IMPLEMENTING THE FPGA DESIGN

Though programming FPGAs traditionally requires detailed RTL and target knowledge, for this design example, we demonstrate the use of an "Algorithm Development" paradigm. To program the target FPGA prototyping platform, we used the design component libraries provided by BPS and thus avoided implementation details such as timing closure and input/output interface issues.

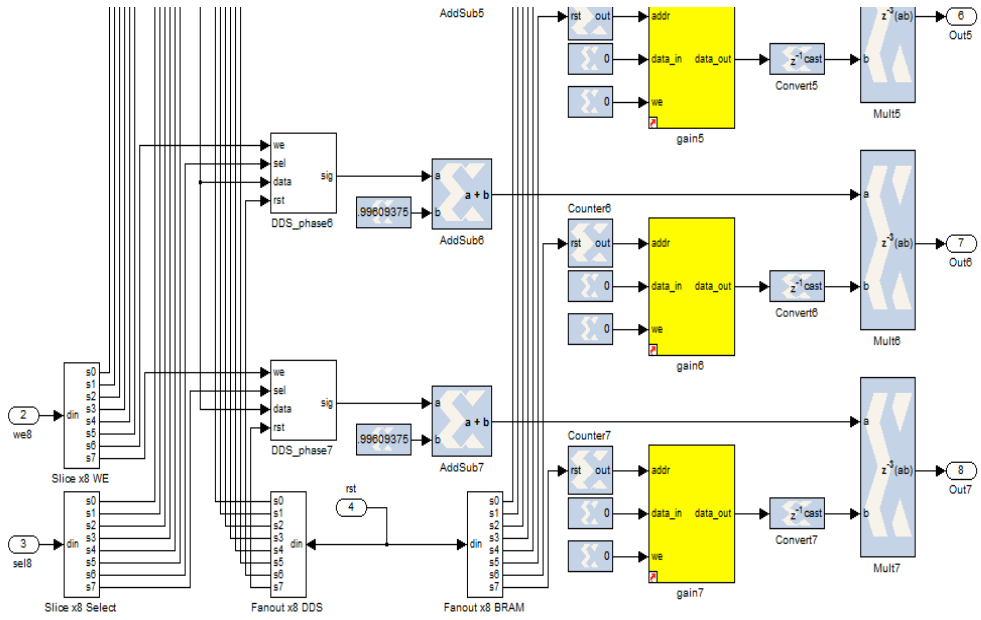

Figure 2: Details of signal generator

4.1. The DAC Signal Generator Design

BPS allows the use of prebuilt libraries to create a DAC signal source model and automatically program the FPGA and all the necessary I/O interfaces. The DAC design used the reference design shown in Figure 1. For this design both the User IP clock rate and the Reference clock rate are set to 250 MHz, 1/8 of the 2 GHz sampling clock rate.

The two BPS Software Register blocks labeled "data" and "ctrl" are used to program the Xilinx DDS (Direct Digital Synthesizer) blocks, which generate the digital sine wave that drives the DAC. The "data" signal is sent to all 8 DDS components and provides the sine wave data. The outputs of the Sine Generator block directly drive the inputs of the DAC block. The output of the DDS component is multiplied with the gain value before being sent to the block output, as shown in Figure 2. The gain BRAM blocks allow for signal modulation up to 2,048 samples.

The BPS DAC block provides a direct, unmodified interface to the data inputs of the DAC device. Because the FPGA to DAC physical interface uses DDR (double data rate) signaling, two full sets of inputs are provided: one set for data to be driven to the DAC on the rising clock edge, and one set for data to be driven to the DAC on the falling edge. Within each set of inputs, each data port corresponds to a value that will be driven by the DAC on its analog output in sequence.

4.1. The ADC Signal Capture Design

The outputs of the ADC block are connected to the "Scopes" block. The "Scopes" block is a wrapper for the software simulation scopes. There are two scopes, "Data" and "Out of Range". The "Data" scope gives a representation of the output of the ADC block. Note that the first input of the "Data" scope bypasses the ADC block allowing comparisons between the source sine wave and the ADC output. The "Out of Range" scope shows when the Out of Range outputs on the ADC block are high, indicating the data is out of the range of the ADC hardware. Again the original sine wave signal, which bypassed the ADC block, is given as reference. See an example of the ADC virtual scope model in Figure 3.

5. RUNTIME SOFTWARE CONTROL

The previous sections have described all the infrastructure provided by the BEE3 hardware platform and the BPS design environment which can be used for prototyping SDR applications. The final component of a complete SDR system is the runtime software control interface itself, which provides the flexibility to monitor and modify the system in the field.

This runtime software interface is provided by Nectar OS, which runs as a lightweight embedded application directly on each FPGA. BPS automatically generates a small microcontroller with attached local memory to each design. This processor is responsible for running the command shell (available via the integrated serial UART in BEE3), performing all initialization and

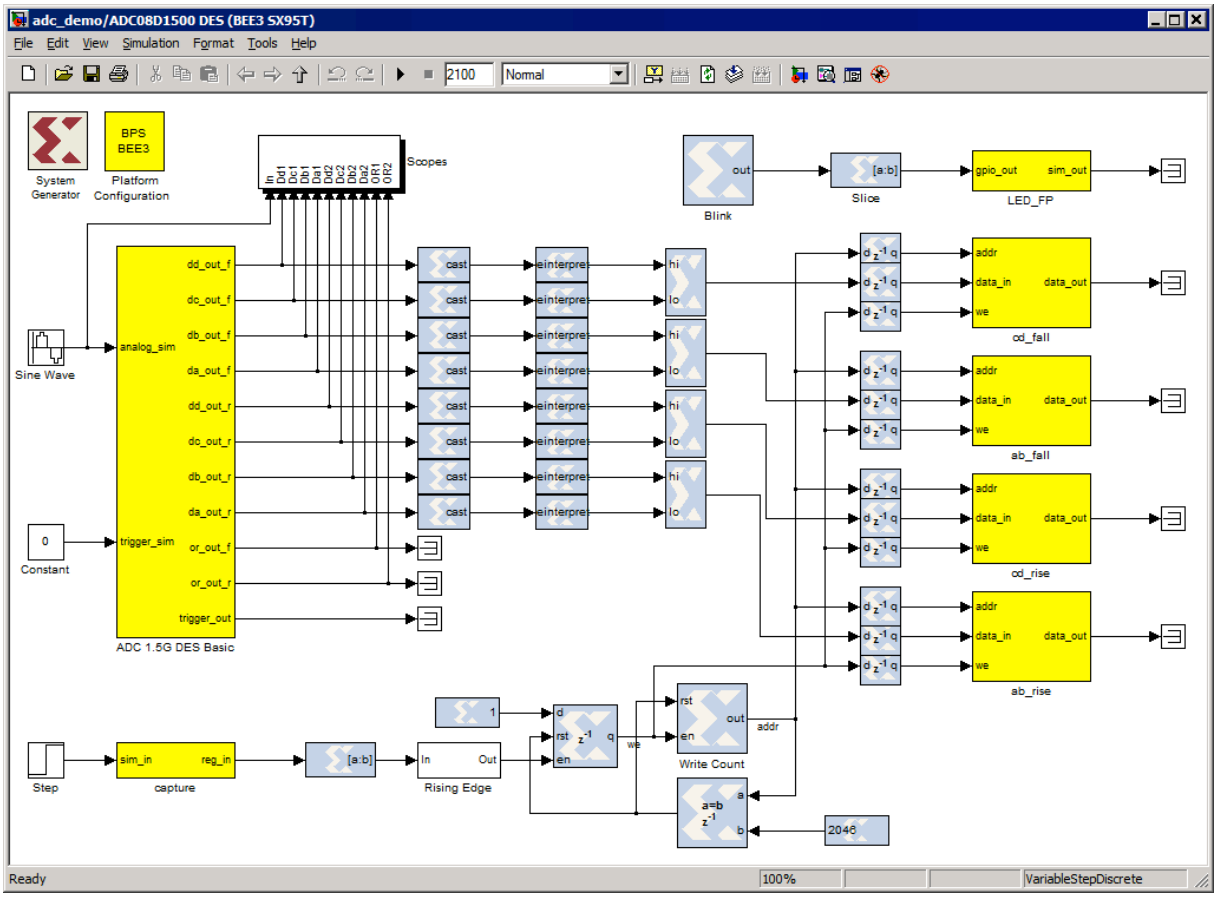

Figure 3: Details of ADC Virtual Scope

maintenance routines defined for specific cores, and optionally servicing all network requests from remote hosts.

In this reference design, software control is provided via shared memory components from the BPS blockset. For the transmitter (shown in Figure 1), two software registers *data* and *ctrl* are used to pass frequency and phase values into the DDS components which generate the carrier tone. A single custom command, *dac_set_freq*, is added to the shell which automatically calculates and sets the appropriate values to drive a specific frequency at the DAC output. This same command can also be implemented on a remote control host, such as VxWorks, and sent over Ethernet as well. The shared BRAM components *gain0* through *gain7* can also be loaded with any arbitrary modulation pattern, which will automatically be mixed onto the carrier tone.

For the receiver, a *capture* software register is used to trigger one window of ADC samples to be captured into local memories upon a rising edge of the register value. These shared BRAM components used to store the samples can then be read via Ethernet at high speed to continuously monitor and analyze the transmission channel.

6. DAC/ADC SIGNAL AND SCOPE PERFORMANCE

In our example design, our DAC Signal Source (FPGA-A) has a system clock of 250MHz, enforced by a DAC sample rate of 2GSps, DDR signaling, and 4:1 mux ratio. The NCO is implemented using Xilinx DDS cores (v2.1) with up to 20 bits of signal output precision, SFDR up to 120 dB and Δf down to 0.02 Hz. The total FPGA (SX95T) utilization for our NCO of 9 bit output, 1Hz resolution, 54 dB SFDR is:

- 4% DSP48s,
- 21% BRAM, and
- 21% Slice utilization

For our ADC virtual scope performance, the FPGA-B system clock runs at 375Mhz, for a sample rate of 3 GSps, DDR signaling, and 4:1 demux ratio. Our total FPGA (SX95T) utilization is:

- 1% DSP48s,
- 16% BRAM, and
- 15% Slice utilization

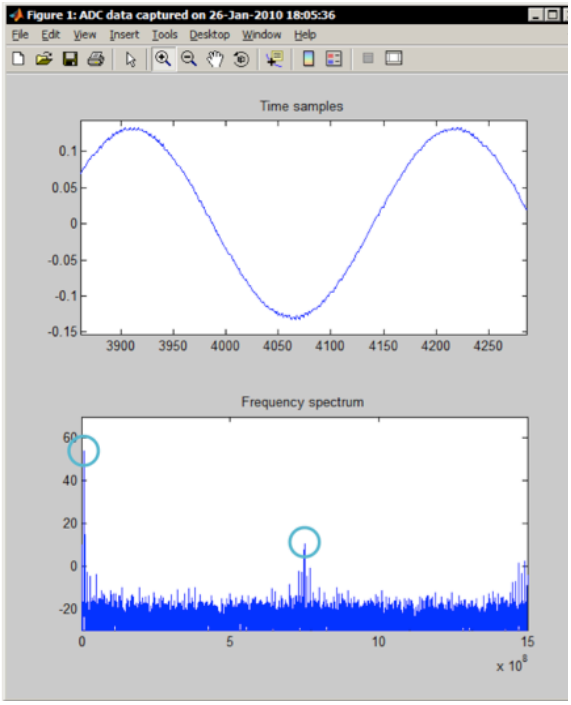

Figure 4: Generated 10MHz Signal

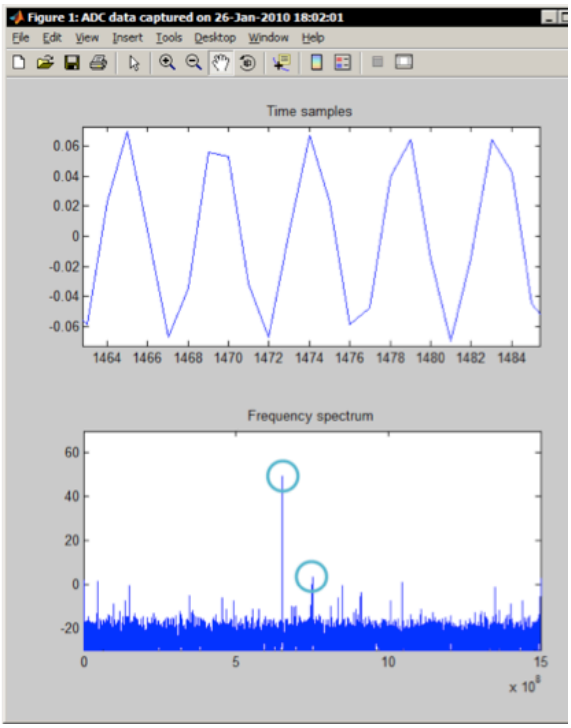

Figure 5: Generated 667MHz Signal

The overall system performance is measured for both a 10MHz tone and 667MHz tone. Results are shown in Figures 4 and 5 and summarized below.

10MHz tone:

- Carrier peak @ 53.9 dB
- Largest spur @ 10.6 dB
- SFDR = 43.3dB

667MHz tone:

- Carrier peak @ 49.1 dB
- Largest spur @ 3.1 dB
- SFDR = 46

7. CONCLUSION

The implication for such real-time/high-speed prototyping systems is essential to adding "intelligence" to signal handling and information, and additional needs exist in the areas of defense applications, including MIMO (Many In Many Out) communications radar applications, signal intelligence and warfare.

8. REFERENCES

[1] BEEcube, Inc. "BEE3-W". http://www.beecube.com/products.

[2] Wind River. "Wind River VxWorks". http://www.windriver.com/products/vxworks.

[3] The MathWorks. "Matlab – The Language Of Technical Computing". http://www.mathworks.com/products/matlab.

[4] BEEcube, Inc. "Nectar Distributed OS". http://www.beecube.com/technology.

[5] BEEcube, Inc. "BPS 3.6". http://www.beecube.com/products.