Application Note AN2090

VECTOR'SoC: A 1 GHz Vectorial Network Analyzer

By: Robert Lacoste
Associated Project: Yes
Associated Part: CY8C26443

Summary
One of the most useful pieces of equipment, just after the ubiquitous spectrum analyzer, is the RF network analyzer and more specifically the vectorial network analyzer (VNA). This Application Note describes how to create a VNA using, among other things, a PSoC™ microcontroller.

Introduction
Digital “stuff” is fun, but any electronics-addicted guy will one day want to investigate the high-frequency radio area, full of strange circuit behaviors and magic black boxes like frequency converters and mixers. This can be fun too, but it can also be a real nightmare without some basic test instruments adapted to the frequency range used. Unfortunately, RF test equipment is very expensive, even on the used market, exactly because it is more than useful!

One of the most useful pieces of equipment, just after the ubiquitous spectrum analyzer, is the RF network analyzer and more specifically the vectorial network analyzer (VNA). A VNA is quite simple in its principles; just apply a frequency ramp to a device under test, measure the signal at the output of the device, and plot the corresponding gain and phase-transfer curves over frequency. For example, Hewlett-Packard (now Agilent) provides some very good VNAs, but these are still in the $1K range even for 10- to 20-year-old equipment. By the way, on the Agilent web site there are very interesting documents describing the basics of VNAs.[1]

I spent plenty of time looking at the ads and auctions, trying to convince myself that I really needed to buy a full-featured VNA, but I suffered from two issues; lack of space in my basement to put this kind of stuff, and lack of a strong enough business case!

Then arrived the PSoC 2002 Design challenge contest. What a good trigger to decide to try the Do-It-Yourself approach myself. And here is the result; Vector’SoC project (Figure 1), a medium performance but very low cost VNA built around a Cypress MicroSystems CY8C26443 chip.

The prototype is fitted into a pretty 6" x 7" x 1" plastic enclosure. The front panel is simplistic: only the input and output RF SMA connectors and a tri-color status LED. This front panel was printed on a color transparency and glued on the aluminum plane.

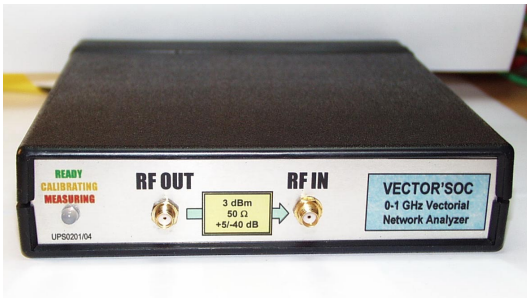

Figure 1: VECTOR'SoC: 1 GHz Vectorial Network Analyzer

[1] Agilent Technologies: AN1287-1, "Understanding the fundamental principles of vector network analysis."

The Vector'SoC is a PC-based instrument. It is connected to a PC through a high-speed 57.6 Kbps serial line. The user interface is friendly Windows-based software, with a real-time refresh of all curves and measurements; which is 2.5 frames/second on my PC.

The Vector'SoC RF performance claims:

- Full coverage from 0 to more than 1 GHz in one band
- Resolution of 50 kHz
- Accurate measurement of gains/losses from +5dB to -35dB over the entire range with a 0.25dB resolution, and up to -45dB from 0 to 300 MHz
- Phase measurement from -90° to +90° with a 3° resolution

All of this with a 100% factory-adjustment free design. Not too bad for a device built with less than $200 of components for the prototype, is it?

These impressive specifications were mainly achieved due to a very simple architecture and use of highly integrated components including Mini-Circuits's RF modules, Analog Devices' AD8302 RF gain and phase measurement chip, and, last but not least, Cypress MicroSystems' CY8C26443 Programmable System-On-Chip device.

Overall Architecture

Figure 2 gives a clear view of the global architecture of the Vector'SoC project as well as frequency and level of the main signals. It is conceptually and physically split into four sub-systems, all separated from each other with good RF shields:

1. First an RF generator module generates the test signal from 0 to 1 GHz. This module is driven by two analog voltages coming from the PSoC microcontroller, one for each of its internal voltage-controlled oscillators (VCO). The first VCO generates an 1160 to 2160 MHz signal, while the second is set to a fixed 1160 MHz. The two outputs are mixed, filtered and amplified, giving a 0 to 1 GHz. Thanks to two RF splitters, this module provides three copies of the output signal; one to the device under test, one used as a reference, and one to a frequency measurement module.
2. The second module is the receiver itself. It gets the reference signal from the generator, as well as the signal that went through the device under test. It scales them, computes the gain and phase delta between the two, and gives them as two analog values read back by the PSoC microcontroller.
3. The third module is a simple pre-scaler used to measure the actual frequency. The frequency measurement itself is done on the PSoC, but this chip is not happy with a 1 GHz signal so a division by 256 helps.
4. Last, the main module, built around the Cypress PSoC chip, takes care of everything else, closing the loop between the generator, the receiver, the pre-scaler and the host PC. It also includes the power supply.

Figure 2 illustrates the global architecture of the Vector'Soc project. The PSoC chip generates two VCO control values. The VCO outputs are mixed to give a 0-1 GHz signal, which is amplified and split into three branches. One is used to measure the actual frequency, the other two are routed to the AD8307 detector chip, one directly and one though the device under test. The outputs of these chips are read by the PSoC microcontroller and sent to the host through RS232.

This architecture is quite exotic, and far simpler than the designs I'm aware of. Usually the RF generator is built around two frequency synthesizers, allowing a very precise frequency from the beginning, but with far higher complexity and cost. I implemented a software-based control loop, using simple low-cost frequency measurement circuitry combined with the power of the on-board mixed-signal Cypress chip. Now let's have a more detailed look at each module.

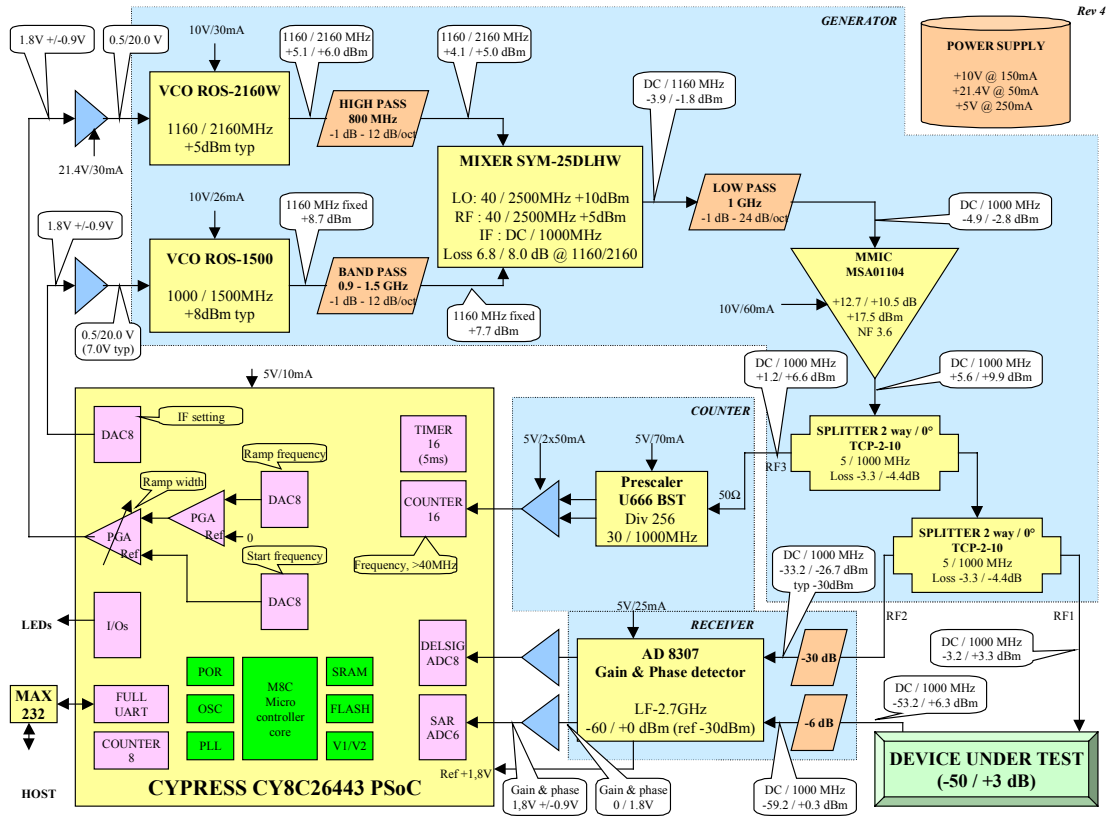

Figure 2: Vector'SoC Global Architecture

RF Generator Module

The RF generator block is mainly built around Mini-Circuits' SMD components. As no VCO can be trimmed down to 0 Hz, two VCOs are needed. The first, V1, is a ROS2160W wide-band VCO, and provides an output signal from 1160 to 2160 MHz when the control voltage ramps from 0.5 to 20 V. The second VCO, a ROS-1500, generates a 1000 to 1500 MHz signal and is automatically adjusted by the firmware to provide a 1160 MHz intermediate frequency. These two signals are then mixed with a SYM-25DLHW integrated mixer, providing a 0 to 1000 MHz signal with a -3.5dBm level.

In order to reduce all spurious frequencies as much as possible, all inputs and outputs of a mixer must go through filters. In particular, these filters block any signal going from the mixer back to the VCOs, as these signals may be "mixed" within the VCOs by any non-linear component, and produce what resembles more a "Christmas tree" on the spectrum analyzer than the pure signal we are looking for.

Here I have tried to stay simple but at least block any major unwanted signal. The difficulty is in the balance between the selectivity of the filters and the difficulty in implementing them properly. At gigahertz frequencies, any parasitic capacitance or inductance may drastically change the filter response from what we designed it for. I decided to use an 800 MHz high-pass filter after the main VCO, a 0.9 to 1.5 GHz band-pass filter after the IF VCO, and a more selective 1 GHz low-pass filter after the mixer. All these filters were designed thanks to a very good online filter calculation tool found on the Internet.[2]

[2] Online Interactive Filter Design, http://www-users.cs.york.ac.uk/~fisher/cgi-bin/lcfilter.

The output of the mixer is amplified with a wide-band MMIC integrated amplifier, a MAV11 also from Mini-Circuits, providing a stronger 8dBm signal (for those that aren't yet fluent with dBm's, this is 6.4 mW over 50 ohms, or 580 mV RMS. More details in Table 1). This signal is then split two times, thanks to two TCP-2-10 miniature 2-ways 0° splitters, providing one output at 4dBm and two outputs at 0 dBm (or 225mV RMS).

Table 1: Basic dBm/V/W Conversion Rules

Voltage (V, RMS)	Power (W)	Power (dBm)
U	$P = U^2 / R$ $= U^2 / 50$	10 . log (P / 1mW)
1.00 V	20 mW	13 dBm
0.225V	1.0 mW	0 dBm
71 mV	0.1 mW	-10 dBm
0,71 mV	0.01 µW	-50 dBm
Voltage (V, RMS)	Power (W)	Power (dBm)
U	$P = U^2 / R$ $= U^2 / 50$	10 . log (P / 1mW)

On my prototype, I've built the entire RF generator module on a double-sided PCB (Figure 3). As usual, in RF designs the bottom side is nearly a 100% ground plane, and all components are surface mounted. Whenever possible the widths of the PCB tracks are calculated for a 50 ohm characteristic impedance (around 1.3 mm wide with a FR4 substrate).

Figure 3 shows a closer look on the RF generator module. The two VCOs are on the left, with the RF mixer on the middle. The output section (MMIC amplifier and the two power splitters) is on the right. Note the RF pass-through filters on the left, used for all low-frequency signals and powers.

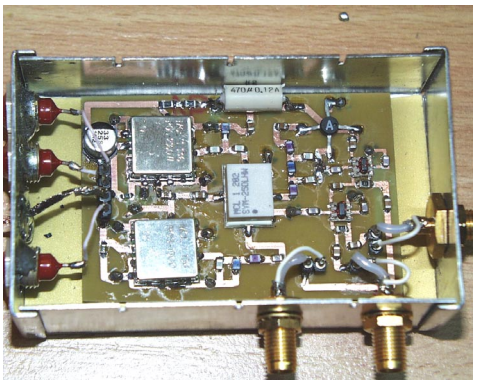

Figure 3: RF Generator Module

The generator PCB is fit into a small-shielded enclosure, with SMA connectors for all RF signals and good pass-though capacitors for all power supplies and control signals.

An AD8302 On-Board

Now let's have a look at the receiver circuit. This circuit receives one of the 0dBm outputs from the generator (reference signal), and the same 0dBm signal that went through the device under test.

This module is quite exclusively built with a single chip, an AD8302 from Analog Devices. This chip is in fact exactly what I was looking for this project.[3] It accepts two signals from 0 to 2.7 GHz and from 0 to -60dBm, and provides phase and gain signals as two analog 0 to 1.8 V outputs. Internally, this chip includes two logarithmic amplifiers, that are then subtracted and multiplied to calculate phase and gain.

As the reference signal provided to this chip must be around -30dBm, I added a small -30dB pad on the reference line (R6 to R8). Moreover, as I wanted my device to accept devices under test with positive gain up to +5dB, and as the generated signal is already at 0dBm, I added a -6dB pad on the main input (R2 to R4).

As for the generator module, the receiver module was built on a double-sided all-SMD circuit board (Figure 4).

In Figure 4, the receiver PCB has only one active component: the AD8307 gain and phase detector. The two RF inputs are on the bottom and are routed to the detector chip through an adapted attenuation network. The bottom layer is a complete ground plane, grounded, thanks to the two copper straps on both sides.

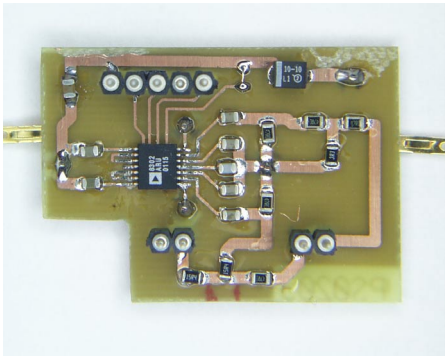

Figure 4: AD8302 Receiver

[3] Analog Devices: « AD8302 2.7GHz Gain and Phase Detector Data Sheet »

Frequency Measurement

With the frequency measurement module, the +4dBm signal is applied on a monolithic U666B pre-scaler from Telefunken. This chip is quite hard to find, but is widely used in old TV-tuners. It accepts a 30 MHz to 1 GHz signal and generates a F/256 differential signal. I added a small TL712 high-speed comparator to transform its differential outputs into a TTL-like signal, compatible with the PSoC MCU inputs.

This module was supposed to be very simple, but took me a very long time to debug. I don't know if my chip is 50% defective or if my PCB design is inadequate, but its operation was quite erratic. Strangely, it worked well when I inadvertently pulled the power supply from 5 V to 5.6 V! Well, quite well, at least now it gives good results from 50 MHz to 750-800 MHz. I still don't understand what was happening there, but I derived a small 5.6 V power source from the power supply and closed the box. This will, of course, need to be solved before any serious industrialization but was adequate for the prototype. As we will see later, the frequency limitation of this pre-scaler is circumvented by software-based interpolation.

Physically, the pre-scaler module is fit into the same-shielded box as the receiver module (Figure 5), but in separate compartments. As for the generator module, all RF connections are SMA connectors and all power and low frequency signals used pass-through capacitors to maintain the shielding.

Shown in Figure 5, the receiver module is split into two compartments. The right-hand side is the receiver itself, built around the AD8302 chip, while the left part is the frequency pre-scaler used to measure the actual output frequency.

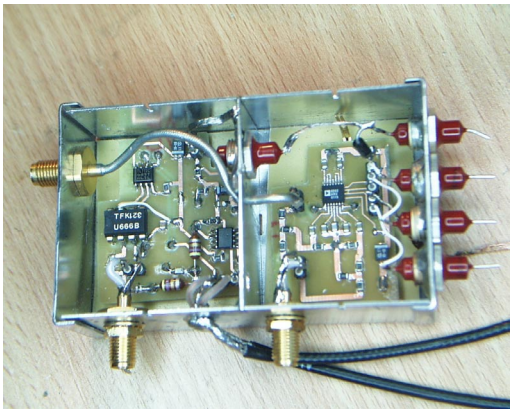

Figure 5: Two Components, Receiver and Pre-scaler

The Main Controller Board

The main module is the only non-shielded module, as it isn't managing any RF signal directly. Thanks to the very high integration of the Cypress PSoC chip, this module is very simple. Around the PSoC CY8C26443, I used two Texas-Instruments TLE2142 dual low-noise operational amplifiers to scale the DAC outputs to the 0-20 V level needed by the VCOs and to convert the 0-1.8 V outputs of the AD8302 chip into the proper levels for the PSoC device inputs. A MAX232 is used to implement the RS232 serial port to the host.

The microprocessor is clocked by an external 32.768 kHz crystal. The internal 2.5% clock reference was not enough to insure precise frequency measurements. This crystal is internally multiplied to provide the basic 48 MHz clock source used by the Cypress chip. As the same pins are used for both the crystal connection and in-circuit programming, some care should be taken on the PCB to minimize track lengths and parasitic capacitance. They were the root cause of a difficult night when I started to debug this project!

Figure 6 shows the opened prototype. The two RF shielded modules are directly behind the front panel, interconnected thanks to two standard SMA jumpers. The main control module is the only non-shielded module. It includes power supply, PSoC chip, RS232 driver and a couple of low frequency operational amplifiers used to bring the PSoC analog signals up to the required levels of the RF module, and back.

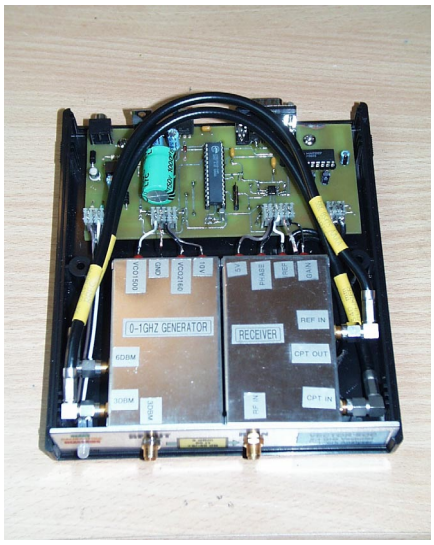

Figure 6: Vector'SoC Open Prototype

One note on voltage references. In order to drastically minimize all drifts in temperature, I configured the CY8C26443 chip in “external AGND/VREF mode” meaning that all its analog signals (including ADC/DACs analog values) are centered around the value of external signal AGND, and have a range from AGND-VREF to AGND+VREF. I derived AGND and VREF from the +1.8 V reference built into the AD8302. This way, even if there is a drift in the 1.8 V reference, there will theoretically be no change in the ADC readings at all, relative to this 1.8 V reference.

The main board also includes the power supply, which is not trivial, as several voltages are needed in this design, such as +5 V for the logic, +10 V for the VCOs, and +21 V for the analog amplifiers in order to get up to 20 V on the VCOs inputs. Moreover, the total power consumption is significant (around 6W), quite exclusively due to the RF components. I ended up with a design built around an external plug-type 12V/1A transformer, two linear regulators for the +5 V and +10 V outputs, and a miniature 12 V to 12 V DC/DC converter with its outputs referenced to the +10 V main outputs, providing +22V. Figure 6 shows the assembled main board integrated with the two previously described RF modules.

Configuring the PSoC MCU

Now we have all the hardware, but we still have to configure the PSoC chip in order to do what we need it to do. Figure 7 shows a logical view of the different modules configured inside the chip. First the clocking section: the two internal counters, 24V1 and 24V2, are configured to provide a 300 kHz clock from the main 24 MHz clock. This 300 kHz clock is used as the main clock for all switched capacitor analog modules within the PSoC, including DACs and ADCs. It is also routed to a 16-bit timer module (TIMER16_1), configured to provide a 5 ms timing period used by the embedded software as a window for frequency measurements. Lastly another 8-bit counter, COUNTER8_1, divides the 48 MHz clock to supply a baud rate clock for the 57600bps UART. As the UART module must be clocked at 8 times the UART speed, this counter is configured to divide the 48 MHz clock by 104 (48 MHz/104 = 8x57600, +/-0.1%). The UART is the full-duplex 8-bits UART module provided in Cypress’s library.

Figure 7 shows the internal configuration of the PSoC for this design. A 300 kHz clock is generated to drive the analog blocks, while two other counter/timers are used to generate the UART clock and a 5 ms timer used by the firmware. Another counter is used to measure the pre-scaler output frequency. Three DACs and two programmable amplifiers output the VCO control signals. Finally, two ADCs (one 8 bit and one 6 bit) read the gain and phase.

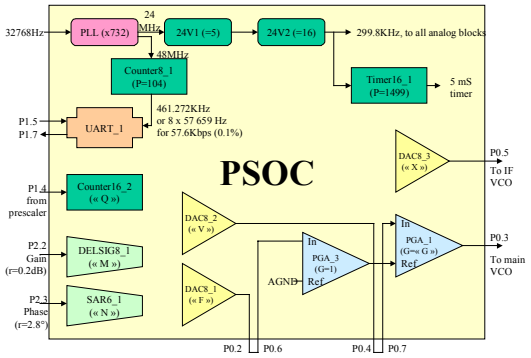

Figure 7: PSoC Chip Internal Configuration

To complete the digital section, another 16-bit counter is configured, COUNTER16_2. This counter is used to measure the output frequency of the RF generator. It is clocked by the pre-scaler output and enabled during a 5 ms window by the embedded software.

Now let us have a look at the DAC section. Two analog outputs are needed; one to drive the main VCO and one for the fixed 1160 MHz IF VCO. For this last one I could have used a fixed potentiometer but I wanted a design without any manual adjustment; I wanted an automatic IF setting feature. Because 8-bit resolution was enough for the IF DAC, I simply used an 8-bit DAC from the Cypress library (DAC8_3). However, the design for the main VCO driving is more complicated and involves two DACs (DAC8_2 and DAC8_1) and one programmable gain amplifier (PGA_1, more on the second PGA later). The principle is as follows, one DAC (DAC8_1) is used to set the lower frequency, while the second DAC (DAC8_2) ramps through the desired frequency range. The PGA gain is dynamically set by the firmware to provide the required frequency scan width. With this design, the effective resolution of the VCO control voltage is 16 bits for narrow band scans, and the design still uses only 8-bit DACs, thanks to the highly programmable nature of the PSoC chip!

Lastly, two ADCs are implemented still within the PSoC to read the receiver gain and phase signals; one 8-bit delta-sigma converter (DELSIG8_1) for the gain measurement, and one 6-bit successive approximation converter (SAR6_1) for the phase.

I must admit that this design was not my first design attempt, but it is the result of an iterative process with a simple goal; find a design that both satisfied the requirements of this project and fit within a single PSoC chip.

Figure 8 shows the internal PSoC placement and routing. This step was not easy, even with the marvelous design tools provided by Cypress, due to the very high usage rate of the chip's resources (more than 90%!). I finally found a solution with an additional PGA block configured as a unit gain buffer (PGA_3), used only as a relay between cells to get a routable design. I also had to route some pins out of the chip and in again to satisfy routing constraints. Conclusion, route the chip first and then design your PCB. This way you will not need some "cut&straps" like the one present on my prototype.

Figure 8 shows the PSoC internal design, as done under the Cypress's designer tool, PSoC Designer. All the digital blocks are used, as well as 10 out of the 12 analog blocks. That's a 90% usage rate, and some headaches to find a block placement that was compliant with the routing constraints of the chip.

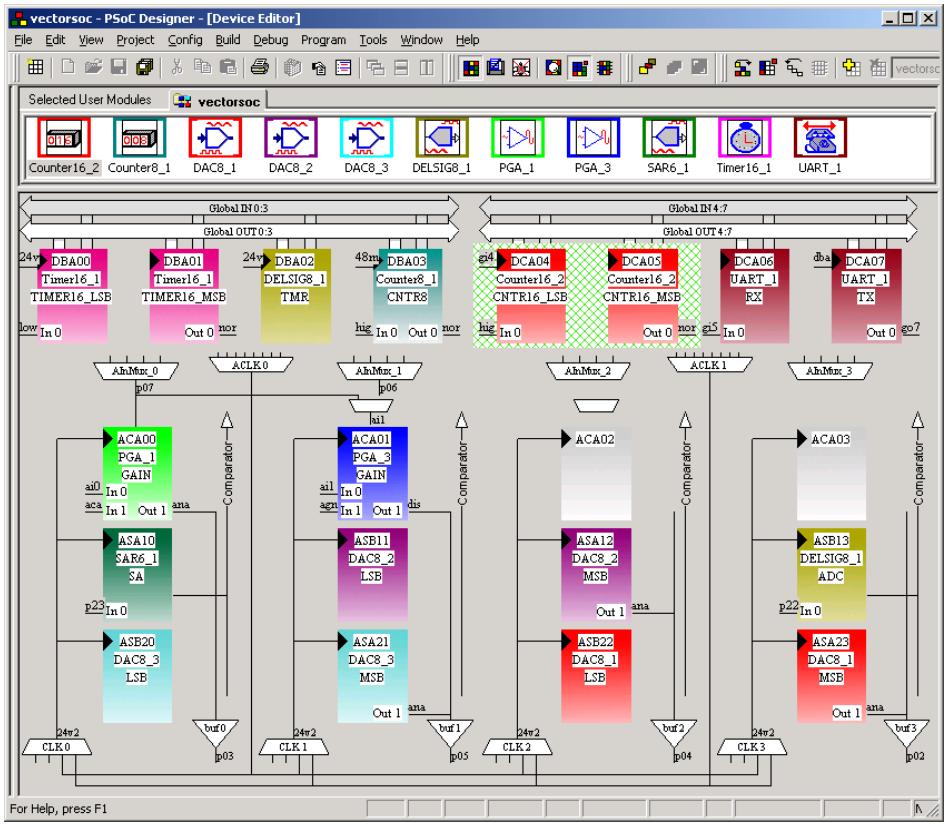

Figure 8: Resource Placement in PSoC Designer

Some Embedded Firmware

In order to have a very flexible design I decided to minimize the features executed in the microprocessor itself, but to do the majority of the work on the PC side. So the embedded firmware is quite simple. The Vector'SoC device is a slave, and simply executes orders sent by the PC through the serial port. The command set includes setting DAC output values, reading the ADCs, reading the output frequency value (by measuring the number of pulses from the pre-scaler during a fixed 5 ms window), and setting the PGA gains, etc. During the optimization of the design, a "combo" command was added to improve the performance, enabling the autonomous generation of a 256-steps frequency scan in one command. This command significantly reduced the overall refresh time (from 1 s down to 0.4 s).

The overall embedded firmware requires only a little more than 2 Kbytes out of the 16 Kbytes of onboard Flash memory. Out of these 2 Kbytes, only 572 bytes are my application itself and the remaining are object images of the libraries provided by Cypress.

PC-Side Software

The last part of this design is the PC-side software. I developed a dedicated Windows-based application under the MFC framework, using Microsoft's Visual C++ 6.0 environment. Figure 9 shows what the user interface looks like. On the right, controls allow designers to select the frequency start and width, as well as the gain scale. Then a mode selector allows designers to launch one acquisition only or to switch to "real-time" refresh mode (around 2.5 refresh/second). Two graphs show the gain and phase response over frequency, while the last plot gives a polar view of the same information (gain and phase graph on the complex plane), with a polar or smith-diagram format. Lastly, a "cursor" feature allows designers to display precisely the gain and phase at a given frequency.

As the embedded software is quite simple, the PC-side software was not trivial. The software first automatically set the IF frequency in order to have a minimum frequency as close to zero as possible. It then enters into a calibration mode, by doing a full frequency ramp from 0 to 1 GHz, measuring the actual frequency each 256 steps. As the pre-scaler chip is not working under 100 MHz and above 800 MHz, a cubic interpolation is calculated with the mean-least-square method.

Two more RF calibration steps are then performed, first with a direct cable between RF input and output (closed loop calibration, used as a reference) and then with 50-ohms loads on both inputs and outputs (open loop calibration, used to automatically calculate and display the noise floor of the Vector'SoC device).

It then executes frequency scans with the parameters set by the user, calculates the effective corresponding frequencies (using the frequencies measured during the scan if they are "reasonable" or the interpolated values), corrects the measured gain and phases using the closed loop calibration values, scales the result and displays it. And does it 2.5 times per second.

Figure 9 shows the Vector'SoC output window with a direct connection from the RF output to the RF input. After a proper calibration, the measured gain stays very close to 0dB as well as the phase, indicating very good stability. The dashed line shows the signal level that was measured during the open loop calibration. The dynamic range is around 40-45dB up to 300 MHz, and is better than 35dB up to 1300 MHz!

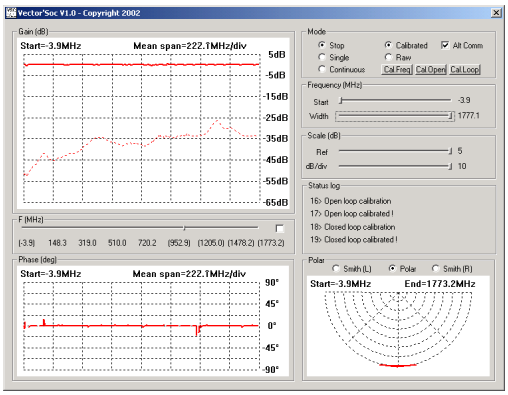

Figure 9: Vector'SoC Output Window in the PC-Side Software

Development Process

Completely developing such a complex project before any test is never a good idea, except for those who like to spend nights trying to understand what is going wrong. Here, I started by developing and building the RF generator module alone and validated it with a good spectrum analyzer. The results were good even without any adjustment in the component values. Then, I developed the RF receiver circuit as well as the pre-scaler module and validated the entire radio chain, end-to-end, using a ramp generator to drive the main VCO and a standard oscilloscope as the output device. This way I have in fact built a full-featured analog-only network analyzer, and this allowed me to validate the whole RF side without any digital modules (Figure 10).

Figure 10 shows what I got on my old trusted Tek 7834 scope when I tested the two RF modules alone, with a ramp generator. The device under test was a simple 4.7 pF serial capacitor, acting as a 400 MHz low-pass filter. The top trace (gain) clearly shows the profile of the low-pass filtering, while the bottom trace (phase) shows the phase sign change at the cut-off frequency.

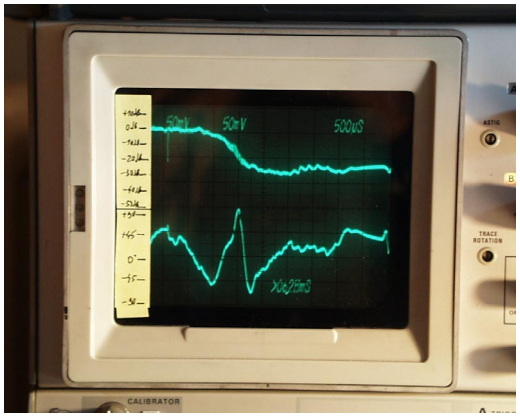

Figure 10: Device Test Results

As this step was more than encouraging, I then developed the main controller board and the PSoC firmware. I decided to use only ASCII characters on the host-to-Vector'SoC serial link in order to be able to fully debug the embedded side without any specific software on the PC (just the classic Hyper-terminal and some macros).

Here, the in-circuit programming feature of the PSoC was very helpful to quickly get working code. The last step was the development of the PC-side software, but this was "only" software in front of proven hardware and firmware, even if I had to fight longer than planned with MFC's strange behaviors, at least for an embedded oriented guy...

Conclusion and Future Evolutions

As shown in Figure 11, the performance of the Vector'SoC device is more than good. It is able to very accurately measure and display frequency responses as narrow as a SAW filter, and give a very accurate and stable display thanks to the automatic calibration routines. The real-time refresh mode is extremely convenient, giving quite the look and feel of a $10K vectorial network analyzer, even if the RF performance of Vector'SoC is, of course, not in the same range.

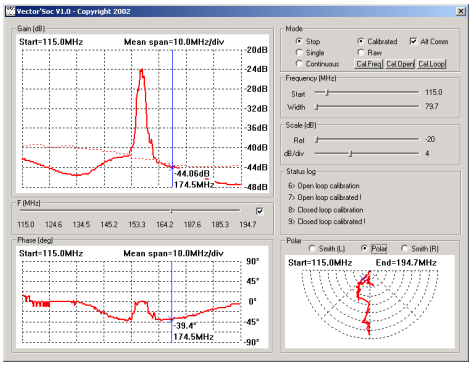

Figure 11: Vector'SoC Performance

The weakest part of the current design is the pre-scaler. Its limited working range and noise susceptibility are limiting factors to the accuracy of the Vector'SoC in very low and very high frequencies. Another chip will be needed, perhaps to directly measure the frequency of the two VCOs before mixing.

On the software side, a very useful improvement would be to implement automatic S-parameter calculation. In fact, the Vector'SoC is already able to measure all S parameters with the use of an external coupler and successive cabling configurations (see Figure 12). A new PC-based application could, however, drive the user in these different steps and directly draw a full transmitted/reflected S-parameter analysis for a specific device under test. A final improvement would be the addition of a phase-group delay graph.

In Figure 12, the Vector'SoC device can be used to measure all RF parameters of a dipole, not only its transmission coefficients. For that, an external directional coupler is needed. External attenuators are also mandatory to measure an active device like a power amplifier.

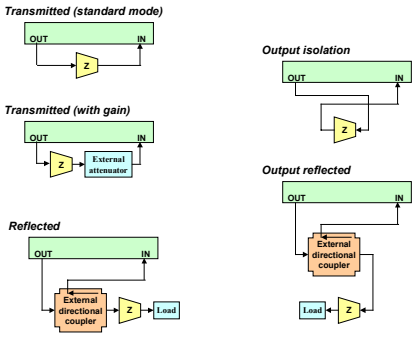

Figure 12: Vector'SoC Measuring S Parameters

Anyway, a project ending with new ideas for improvement is always a good project! And in this case, the current Vector'SoC prototype already provides very useful test capability for the RF enthusiasts as well as a very good example of the effective use of highly integrated mixed-mode devices like Cypress's PSoC MCU!

I hope you've enjoyed reading about the Vector'SoC project as much as I have enjoyed developing it. It was fun!

Sources

- TCP-2-10 splitter, ROS-2160W & ROS-1500 VCOs, SYM-25DLHW mixer, MAV11 amplifier: Mini-circuits, www.minicircuits.com.
- CY8C26443: Cypress Microsystems, http://www.cypressmicro.com/.
- U666BST pre-scaler: Telefunken electronic.
- AD8302 detector: Analog Devices, www.analog.com.
- TLE2142 low noise operational amplifier, TL712 high-speed comparator: Texas Instruments, www.ti.com.

About the Author

Name:	Robert Lacoste
Title:	Consultant
Background:	Robert Lacoste has 12 years experience in embedded systems and cost-optimized developments. His expertise in mixed-signals and microcontroller-based designs has been recognized by more than 10 prizes in international design contests. He is based in France, near Paris.
Contact:	rlacoste@alciom.com

Cypress MicroSystems, Inc.
22027 17th Avenue S.E. Suite 201
Bothell, WA 98021
Phone: 877.751.6100
Fax: 425.939.0999
http://www.cypressmicro.com/ / http://www.cypress.com/aboutus/sales_locations.cfm / support@cypressmicro.com