Using FPGAs In Automotive Radar Sensors

Georg Hanak, Product FAE
Altera GmbH
Phone: +49 (89) 321825 0 / Fax: +49 (89) 321825 79 / ghanak@altera.com
Freisinger Strasse 3, 85716 Unterschleissheim, Germany

Dr. Ralph Mende, Managing Director
smart microwave sensors GmbH
Phone: +49 (531) 39023 0 / Fax: +49 (531) 39023 58 / ralph.mende@smartmicro.de
Mittelweg 7, 38106 Braunschweig, Germany
www.smartmicro.de

Abstract

Driver assistance systems have become more popular with the introduction of 24GHz radar systems for passenger cars. In addition to lane change assist and blind spot sensors, applications like ACC (adaptive cruise control) and as an extension of that, ACC Stop&Go are being addressed by OEMs and sensor manufacturers. The car manufactures are now focusing their interest on fusion-based multi-sensor systems, enabling the car to monitor the whole environment. One of the requirements resulting from this system set-up is a new distribution of signal processing blocks between sensor(s) and a central ECU. As this fusion-ECU becomes responsible for data validation, object recognition, object tracing and communication with the car network, the sensor itself becomes a simple data acquisition unit. Ideally it would transfer all data to the central ECU for processing. Assuming a typical cycle time of 30-40ms this could result in data rates up to 2.9Mbit/s per sensor. This bandwidth isn't available with today's car networks; therefore some data pre-processing and data reduction has to be performed in the sensor. One approach is to implement the required signal processing in an FPGA. With their internal multipliers and RAM blocks, they offer an unbeatable DSP performance at a moderate frequency. Their architecture is suitable for the whole algorithm including offset correction, FFT and digital beam forming. Even tasks like threshold calculation and spectral peak detection can be performed within the FPGA. On the other side, with the flexibility of an FPGA, the system can be adopted to any available bus like CAN, FlexRay or a proprietary bus without changing the system architecture and adding additional external components. A soft processor core optimised for FPGA can be used as system controller and also as host controller for CAN or FlexRay. Using FPGA, cost of the new multi-sensor systems can be kept low, making the new 24GHz radar systems affordable for customers and thus improving road safety..

I. UMRR Radar Sensor Technology

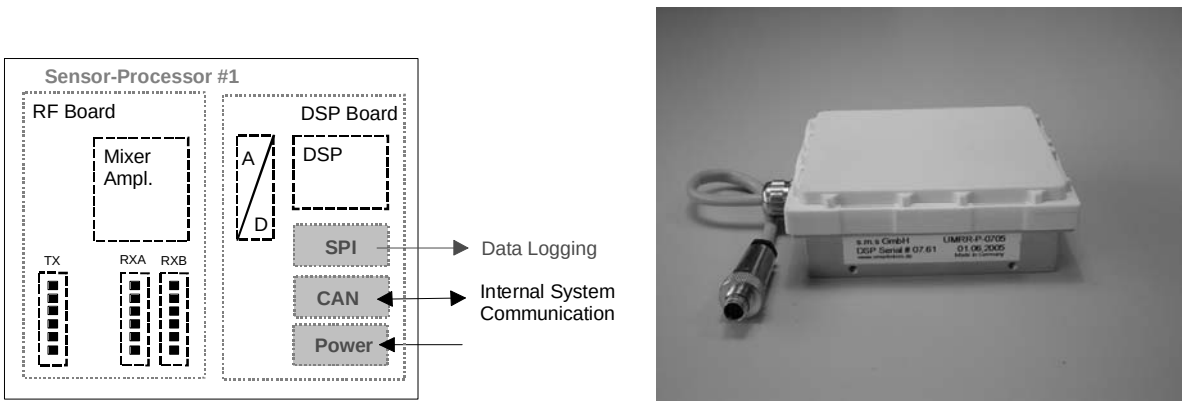

Figure 1: Block Diagram and Photograph of the UMRR Radar Sensor

The name UMRR stands for Universal Medium Range Radar. The design targets of the UMRR sensor platform were mainly performance and flexibility, the intention was to cover a certain range of automotive applications, with the focus on high dynamic scenarios.

Key Performance Facts:

- Direct and simultaneous measurement of range, velocity and angle.
- Short measurement time.
- Good min. range, medium max. range (typ. 50-70m).
- Conformity with ETSI and FCC frequency regulations.
- One-Box-Design with integrated detection, tracking and communication software.
- Stand alone or network (sensor fusion) operation.

A sensor unit consists of two components: RF frontend module and DSP module. This is depicted in the block diagram in Figure 1, a photograph is given on the right. The size of the device is only (including processor) 94x78x31mm (WxHxD).

The RF module contains a VCO, transmit and receive amplifiers, mixers, and is made all planar of discrete components. There are also means provided for the waveform generation as well as for the control of in-band emissions and waveform linearity. A mono-pulse principle patch antenna type (one transmitter, two receive antennae) is applied.

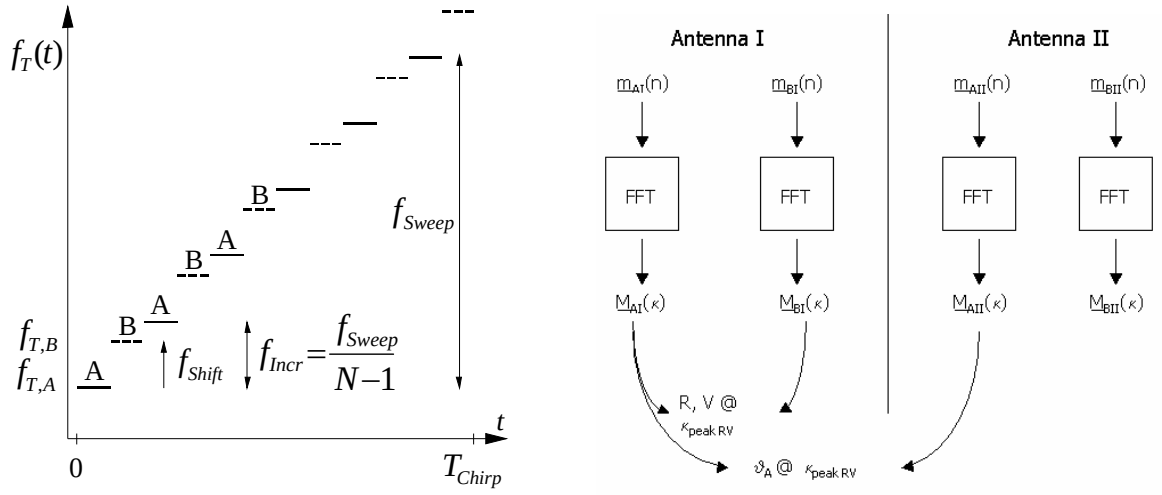

Figure 2: Transmitted Signal and Signal Processing Principle

The narrowband FMCW transmit signal is created by software. As an example, an FMSK signal [1] is depicted. This combination of FSK and LFM waveform design principle offers the possibility of a direct, unambiguous and simultaneous target range, velocity and angle measurement. Only one chirp (up- or downchirp) is run per cycle, please see the simplified block diagram in Figure 2.

The radar is able to resolve (separate), handle and track multiple targets (i.e. radar reflectors), in the following list an overview is given of the tasks and signal processing blocks which need to be executed in the radar.

1. Generation of the transmit signal and sampling of the input data (receive signals, see Figure 3)
2. Offset Correction
3. FFT
4. Threshold calculation

5. Frequency and phase value extraction from the spectrum for selected bins (see Figure 4)

---------- *Low level fusion interface*

6. Calculation of targets parameters
7. ambiguities resolution
8. Target acknowledgement algorithm (see Figure 5).
9. Target tracking

---------- *High level fusion interface*

10. CAN data link to superior control unit (driver warning algorithm or automatic vehicle control loop)

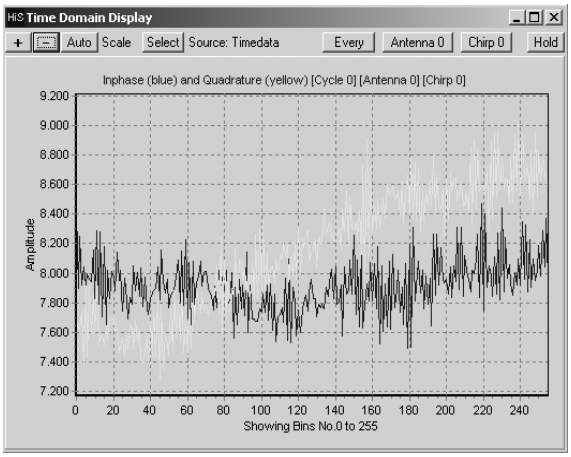

Figure 3: Example time domain signal.

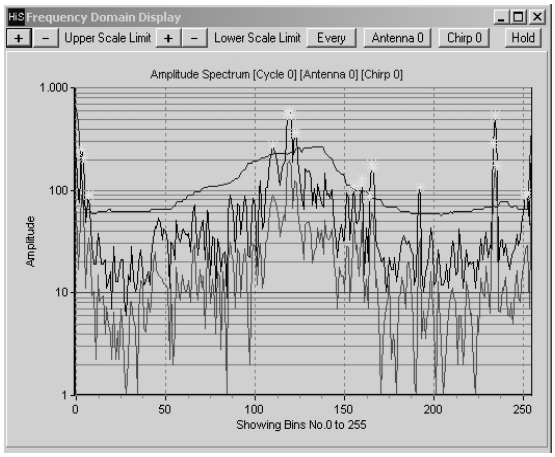

Figure 4: Example frequency domain signal with threshold and selected bins.

Figure 5: Example view of detected targets.

In the table below, a list of technical data is provided.

Parameter	Value
Model	UMRR-P-0708
Operation Principle	FMSK
3dB Bandwidth	< 100MHz
Minimum Range	1.5m, below 1.5m: Presence Detection Available
Maximum Range	70m
Cycle Time	39ms
Velocity Interval	-69.4...+69.4m/s
Carrier Frequency	24.125GHz
Maximum Transmit Power	20dBm
Antenna Type	Patch Antenna
Field of View (Example)	+-35° (Azimut) x 10° (Elevation)
Supply/Interface	9V-36V / CAN, RS232, USB, Flexray, SPI

Table 1: Technical Data

In the actual sensor generation, designed for stand-alone operation, a fixed point DSP is applied. The chip already has A/D converters, a CAN module, Flash and some RAM memory on board and requires just a small number of additional components. The radar makes use of the CAN link and is thus able to deliver data to a superior control unit. In configurations where more than one radar is required, that ECU runs sensor fusion algorithms.

Due to the data rate limitation on that CAN link, a low level fusion would not be possible with this sensor generation.

II. Description of the application.

The example application is related to automotive comfort-oriented driver assistance: **ACC Stop & Go.** This is an extension of the conventional ACC (Adaptive Cruise Control) function.

An off-the shelf ACC sensor typically has a narrow field of view (see Figure 6). Operating at 77GHz, this type of sensor is today in volume production and fitted to a number of high-class models. The limitations are: operation typically only above 40km/h, narrow field of view, late detection of cut-in situations, no full coverage of the driving lane close to the vehicle, poor measurement performance at short range.

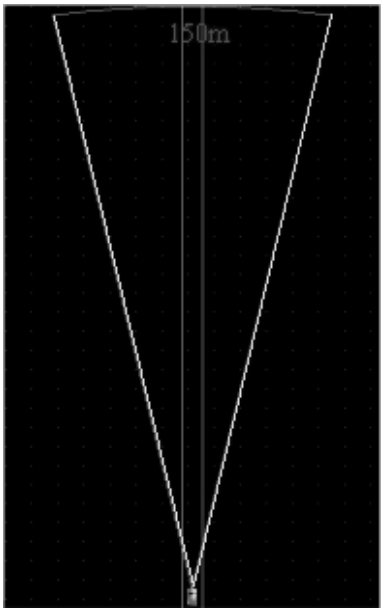

Figure 6: Typical ACC sensor field of view.

To extend the function, in addition to the 77GHz ACC sensor two 24GHz UMRR sensors are placed behind the front bumper fascia, close to the corners. The combined field of view is selected to cover the full width of the car, beginning directly at the car's front bumper fascia. It covers 3 lanes; the outer limits of the field of view are +45° and -45° from the left and right front corner of the car respectively, given that the mounting position and pointing direction are appropriately selected. Excellent short and medium range measurement performance is provided.

Three driving lanes can then be covered; this can lead to a significant improvement of the function of long range 77GHz ACC sensors, when the detection results of all three radars are subject to a sensor fusion. Enhanced features like follow-to-stop or stop-and-go become possible.

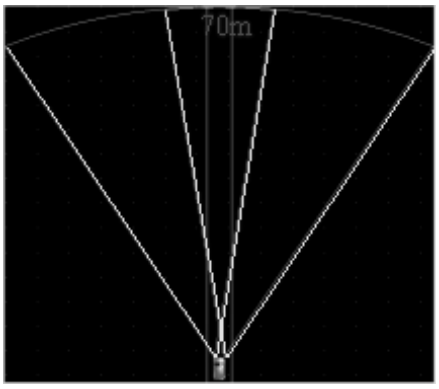

Figure 7: Additional field of view for Stop & Go operation.

Figure 8 depicts the typical signal processing flow of a standard ACC sensor, and the lower two signal paths represent the extension of the two additional 24GHz units. To obtain an optimum sensor fusion performance, a low level data interface is required. For commercial (system price) reasons, the additional 24GHz sensors have to be as small as possible in size, as low as possible in cost. Therefore a customized solution is needed, and this target can be best met using an FPGA solution.

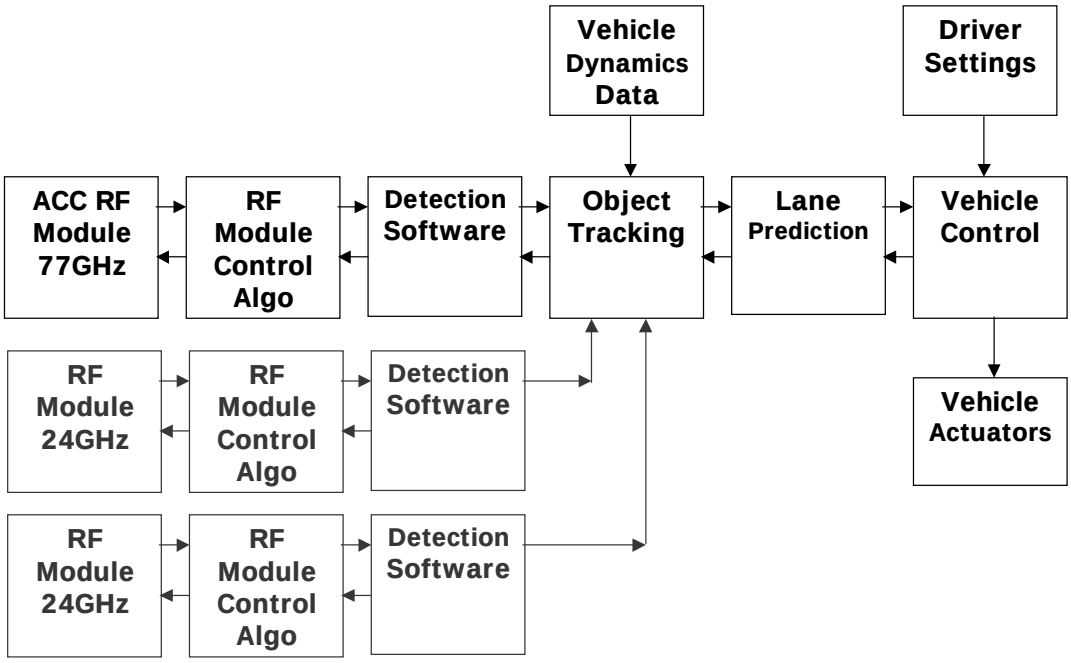

Figure 8: Signal processing flow of a combined ACC Stop&Go sensor system.

III. Altera DSP Tools & IPs

DSP system design requires both high-level algorithm and HDL development tools. DSP Builder integrates these tools by combining the algorithm development, simulation, and verification capabilities of The MathWorks MATLAB and Simulink system-level design tools with VHDL synthesis, simulation, and Altera development tools. The DSP Builder shortens DSP design cycles by helping designers create the hardware representation of a DSP design in an algorithm-friendly development environment. The existing MATLAB functions and Simulink blocks can be combined with DSP Builder blocks and intellectual property (IP) functions to link system-level design and implementation with DSP algorithm development. DSP Builder allows system, algorithm, and hardware designers to share a common development platform.

Designers can use the blocks in DSP Builder to create a hardware implementation of a system modeled in Simulink in sampled time. DSP Builder contains bit- and cycle-accurate Simulink blocks, which cover basic operations such as arithmetic or storage functions. Complex functions can be integrated using MegaCore functions in DSP Builder models. Like the required FFT. See Figure 9.

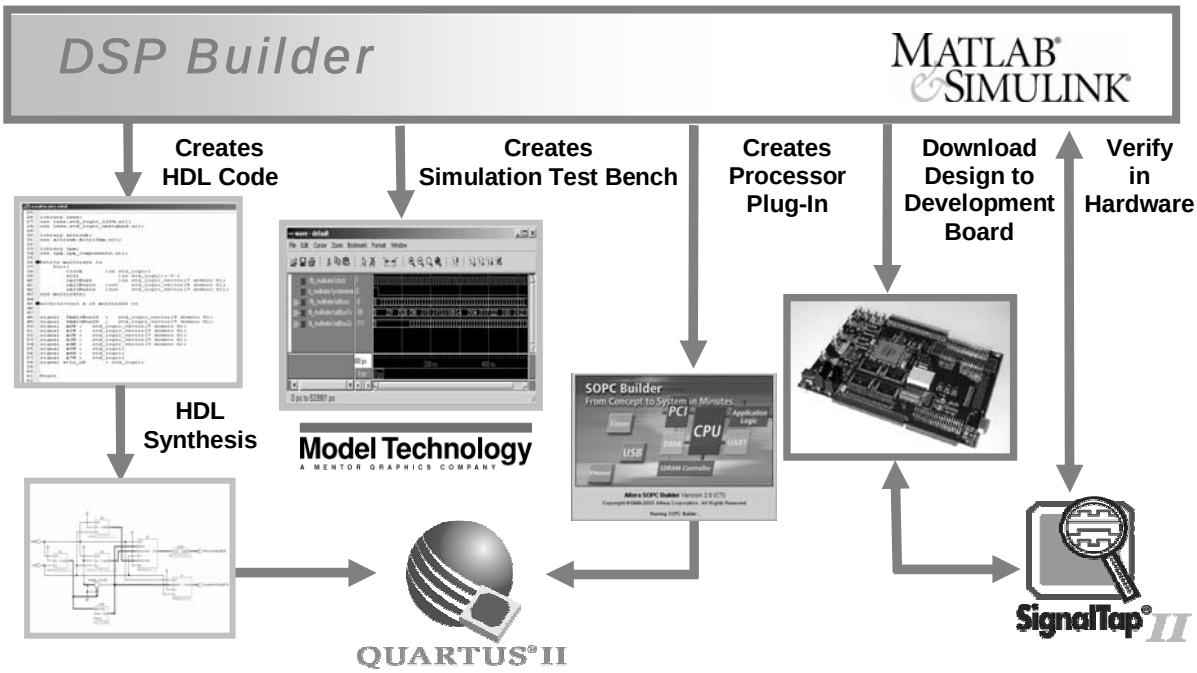

Figure 9. DSP Builder Overview

MegaCore functions support Altera's IP evaluation feature, which allows the designer to verify the functionality and timing of a function also in hardware without purchasing a license.
The DSP Builder SignalCompiler block reads Simulink Model Files (**.mdl**) that are built using DSP Builder and MegaCore blocks and generates VHDL files and tool command language (Tcl) scripts for synthesis, hardware implementation, and simulation.

IV.) FPGA based Automotive Systems with Embedded Processors

Altera Cyclone II FPGAs are optimized for processor-based applications, especially those that benefit from the use of an embedded soft processor such as Altera's Nios II processor (shown in Figure 10). A typical Nios II controller system consists of a CPU, the on-chip RAM and ROM, an external memory controller, and a number of serial and parallel interfaces. All modules are connected to the Avalon bus via a multi-master-capable switch matrix. The Nios II processor is a 32-bit RISC processor and is based on a Harvard architecture that has completely separate data and address buses fully supported by the Avalon switch matrix.

The Nios II processor's synchronous interface and small resource utilization and optimized performance make it particularly well suited for implementation in programmable logic devices.
Users of the Nios II processor also have the option to add custom instructions to their processor designs. These instructions can be defined by the user and implemented in hardware. For example, a multiplication function, which takes up about 80 clock cycles when emulated in software, can be performed in just two cycles when executed as a custom instruction. This makes it possible to integrate a variety of functions such as accelerating line drawing tasks, with the instruction set capable of handling up to 256 different instructions.
The Nios processor design environment consists of a parameterizable hardware description and an automatically-adapted software development environment. Nios II processors can be implemented in Altera devices such as Cyclone II, Stratix II and HardCopy devices. Typical Nios II processor implementations in FPGAs achieve performance levels between 100 and 225 MIPS. Operating systems already ported to the soft processor include ATI Nucleus, µClinux, µC/OSII, and KROS.

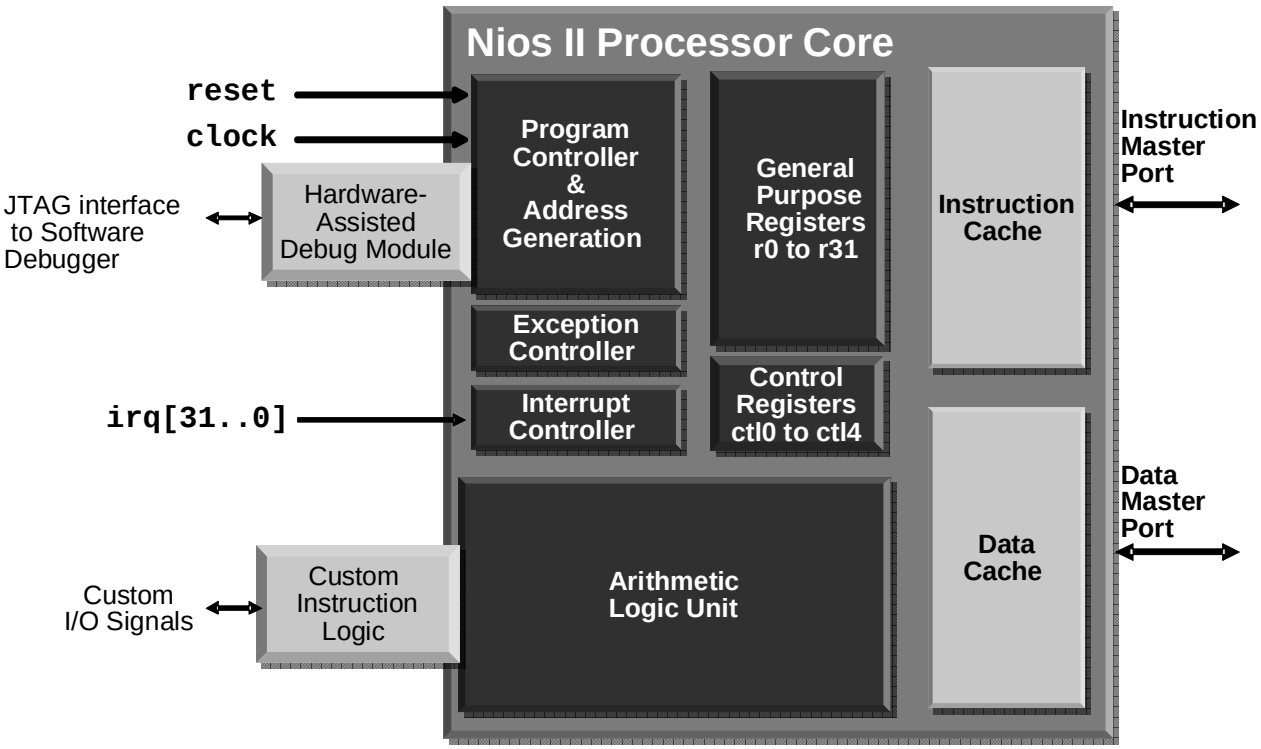

Figure 10: NIOS II Block Diagram

V.) Low Cost FPGA

Following the first-generation Cyclone family, Cyclone II FPGAs were build from the ground up for low cost and provide a customer-defined feature set for high-volume solutions to cost-sensitive applications. Cyclone II devices are manufactured at TSMC's 90-nm low-k dielectric process to deliver high performance and low power consumption at a cost that rivals that of ASICs. It's on the designer to decide if he uses a FPGA alone or as digital signal processing (DSP) co-processor to improve price-to-performance ratio for DSP applications. Cyclone II devices include an optimized set of DSP features and are supported by Altera's full set of DSP flows. Cyclone II DSP support includes:

- 18x18 multipliers
- on-chip embedded Dual-ported memory
- High-speed interfaces to external memory
- DSP IP cores
- DSP Builder interface to The MathWorks' Simulink & MATLAB software

As we have seen in a previous section the presence of multipliers is vital for the application. Therefore the Cyclone II family was chosen for the DSP replacement. To see the real benefit of an FPGA implementation we have to look deeper into the multiplier details. The embedded multipliers in Cyclone II FPGAs are capable of implementing the simple multiplication operation commonly used in typical DSP functions. Each embedded multiplier (shown in Figure 11) can be configured as one 18-bit x 18-bit multiplier or two 9-bit x 9-bit multipliers.

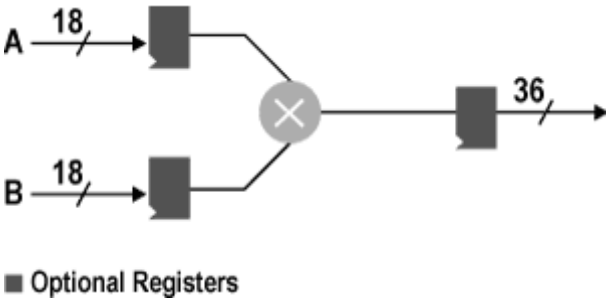

Figure 11. Embedded Multipliers in Cyclone II Devices

The embedded multiplier supports both signed and unsigned multiplication. It also offers optional input and output registers for increased performance. The embedded multipliers are also seamlessly integrated with the embedded memory blocks to provide an efficient implementation of DSP algorithms that uses both multiplication and memory operation.

VI. Radar Algorithms Implementation on FPGA

For most generic DSP functions like FIR filters, FFTs and multiplication intensive data processing the DSP-only approach significantly reduces the implementation effort with the availability of pre-built, assembly optimized, C-callable library functions. Even if the designer requires a custom function that is not available with generic library functions, complex algorithms are generally easier to implement in a high-level language like C or C++. Challenges will arise while trying to optimize the performance of the function for a particular DSP, often requiring an in-depth knowledge of the processor architecture and assembly instructions. Nevertheless, the designer remains within the same familiar development environment, and is not required to build additional hardware functions to complement the desired system.

On the other hand, the FPGA approach currently requires a certain amount of hardware knowledge to assemble the various components of the FPGA co-processing system and partitioning the task into a data path application implemented in hardware and a control application ideally running on a processor, in our case on NIOS II. When transforming the data path application to hardware the designer has to decide which parts shall be implemented using predefined and optimized functions and which parts needs to be implemented using DSP Builder library elements. Altera offers optimized, parametrizable Megacore functions for the required FIR and FFT. All other functions like Offset correction, IQ-Correction and Digital Beam Forming can be easily implemented and simulated using the DSP Builder elements. Having Avalon port elements in the DSP-Builder library allows combining DSP Builder generated designs with SOPC Builder. This again allows implementing the control functionality that would require huge state machines in hardware on NIOS II as a C/C++ program.

VII. Summary

DSP and FPGA solutions provide designers with a myriad of implementation options and solutions for today's system designs. Along with these solutions comes a variety of design factors and considerations that need to be evaluated to select the best approach, depending on system requirements like ease of implementation, cost and performance as well as power consumption. Digital signal processors can provide the simplest implementation for a wide range of DSP algorithms and applications, but the cost/performance, implementation flexibility and hardware parallelism provided by an FPGA cannot be overlooked.

From a price/performance comparison, FPGAs provide better performance for lower cost compared to a DSP approach. Additionally, if the FPGA is not fully utilized, more functionality and parallelism could be added to the FPGA to increase the amount of processing the FPGA is capable of without impacting the cost of the system. Also, from a function-to-function power comparison, we see that for the same function, an FPGA implementation is capable of consuming less power than a digital signal processor.

VIII. References

[1] Meinecke, Marc-Michael; Rohling, Hermann:
Waveform Design Principles for Automotive Radar Systems
German Radar Symposium, Berlin 2000.