Chapter 5
Test application: RADAR signal processor for automotive applications

Chapter 3 discussed some of the optimization directives available in existing HLS tools. The results achieved as a result of the usage of these directives on applications that were created using MATLAB HDL coder and Vivado HLS were also discussed. The implementation results were compared with those achieved from hand-coded RTL-based design flows. In Chapter 4, we proposed a novel optimization method for HLS-based designs named "application-specific bit width for intermediate data nodes." We shared the results obtained as a consequence of applying this methodology to multiple designs. We also presented a comparison of the implementation results for these applications against hand-coded RTL implementation results available in the literature. In this chapter, the learnings obtained from both Chapters 3 and 4 have been applied to a new test application of RADAR signal processor. The FPGA target area and speed of operation was optimized for an automotive RADAR signal processor used in advanced driver assistance systems (ADAS) applications [5.1, 5.2, 5.3]. A combination of directives (discussed in Chapter 3) and the novel optimization method (discussed in Chapter 4) were applied to achieve the desired results after a base implementation was created using HLS. In addition to the usage of HLS methodology for design, a verification framework was also developed to test the functionality of the design with stimulus applied and outputs sampled at a much higher abstraction level making it easier to model real-life traffic scenarios. These scenarios and stimuli helped in thorough and extensive verification of the design under test.

5.1 Introduction to RADAR signal processor application

Very large-scale integration is usually applied across several segments to design chips for household appliances, cameras, mobile phones, etc. Fast progress in technologies such as artificial intelligence and machine learning in the automotive segment has led to a growing number of hardware accelerators being integrated into modern ADAS SoCs [5.4]. In radar-based ADAS SoCs, baseband processing is required to detect the speed, distance and angle of elevation of the target (e.g., vehicle, pedestrian, and traffic sign). The target and the source frequently move at high speeds; hence, the computation rate must be sufficiently high to execute actions (e.g., braking) in real-time. Software-based implementations of such systems or accelerators fall short of the desired performance, which has led to an increment in the demand for custom hardware implementations, e.g., on FPGAs. This is because they offer parallel architectures and hence high speeds [5.5].

Further, the growing complexity and configuration capabilities of such accelerators, as well as the requirement to deliver functionally verified chips to customers, has resulted in certain requirements as listed below:

1) Detection of corner-case RTL defects very early in the design cycle.
2) Support for embedded software development of hardware accelerators as concurrently as possible with the hardware development cycle.
3) Running of a large number of design clocks in a shorter wall clock time during verification process. This is because these accelerators' results are accumulated over many design cycles and cannot be assessed in conventional software simulations.
4) An environment modeling framework that can produce input data corresponding to a wide variety of situations. In this way, it is possible to interface with multiple radar sensors to compute the range, velocity, and azimuth angle for various objects in a vehicle environment.
5) Permitting designers or architects to experiment with several implementation options for the accelerator. For example, if the architect chooses to implement a 128-point fast Fourier transform against a 256-point one and analyze the results, a new design should not be generated from scratch.

Regrettably, the above-described challenges are hard to address in a conventional hand-coded RTL-based design and simulation-based verification environment. Thus, HLS-based area and speed optimal implementations for these accelerators targeting FPGAs is the need of the hour.

In this work, a novel design and verification framework for a RADAR processing SoC is introduced. The framework is supported by an HLS-based design scheme for the processor. Moreover, it supports the application of a real-world stimulus to register transfer-level design implementation running on FPGA. The RADAR processor executes range and doppler processing on the baseband signal, followed by beamforming. All of this is to calculate speed, distance, and azimuth angle for the target in real-time. Applying the framework, the design (RADAR signal processor) is readily modeled in environment scenarios requiring multiple vehicles, environment conditions, channel interference, obstacles, etc. The design is implemented on the Xilinx Kintex-7 FPGA and validated using the FPGA-in-the-loop feature of the MATLAB HDL Verifier [5.6]. The RTL synthesis results are optimized for the area, speed, and power using

HLS directives, and they were found to be superior than the hand-coded RTL synthesis results for the identical application. The functional simulation results are observed as equivalent to field testing results obtained by post-silicon application teams with similar input stimuli.

Customer use-cases for the distance and velocity computations are performed in a pre-silicon environment using range and Doppler processing on the Xilinx Kintex-7(XC 7K 480T) FPGA. The findings reveal that the proposed framework based on MATLAB HDL coder and HDL Verifier is better than other similar implementations from published literature in terms of speed and FPGA resources. This is due to the usage of applicable HLS directives (discussed in Chapter 3) and a new design method based on application-specific bit width for intermediate data nodes (presented in Chapter 4).

5.2 Prior works for RADAR signal processor implementation

Lin et al. employed a new target detection algorithm applying a pairing mechanism and spatial filter design for automotive radars [5.7]. The field measurement results of the frequency-modulated continuous-wave (FMCW) radar baseband processing system designed in their study showed effective execution in a realistic application scenario of various target vehicles around the source. Fegar et al. introduced an experimental verification setup of a 77-GHz synthetic aperture radar (SAR) system for automotive applications [5.8]. They utilized a simplified signal processing algorithm derived by exploiting various approximations valid for conventional automotive measurement settings. Yue et al. introduced a precise measurement of distance and velocity in automotive radars based on the quadratic function method for signal processing [5.9]. Zeng et al. introduced a procedure-based validation method for SAR signal processing algorithms considering the external environment’s confluences [5.10]. All of these designs are based on hand-coded HDL implementation and not using HLS. Moreover for all the above implementations, authors have chosen HDL simulation based approach to verify the design and prove the functionality. Moreover, in terms of functionality, all of these designs have been implemented on FPGA platforms without having an available comparison with real silicon results.

5.3 RADAR signal processor and verification framework

The principle of RADAR is based on determining the range, azimuth angle, and velocity of the target (vehicle) from the received echo of the electromagnetic radio waves transmitted by a transmitter. RADARs are very common in ADAS systems as they can be used under all visibility conditions, unlike vision-based systems. Due to the recent advancements in RADAR signal processing techniques and millimeter-wave (mm-wave) semiconductor technology, automotive radars are commonly used in ADAS systems. Based on their range and the type of application they are applied in, automotive radars are grouped into long-range, short-range, and medium-range radars. Their corresponding use-cases are compiled in Table 5.1

Table 5.1. Radar characteristics based on range

RADAR type	Long range	Short range	Medium range
Range (m)	30-250	1-10	10-30
Azimuth Angle	+/−15°	+/−80°	+/−40°
Elevation Angle	+/−5°	+/−10°	+/−5°
Applications	Automotive cruise control	Lane change assist, Rear collision warning, Cross traffic alert, Blind-spot detection	Park assist, Obstacle detection

Automotive radars are usually intended to obtain target information from reflected electromagnetic waves. This information usually consists of the range (the distance of the vehicle), velocity (relative speed of the vehicle to the source), and elevation (size and azimuth angle of the target) [5.11, 5.12, 5.13].

The transmitted and received signals for an FMCW radar are shown in Figure 5.1. As displayed in Figure 5.1, the bandwidth (BW) is a measure of the radar range resolution, i.e., a high-range resolution needs a transmission signal with higher bandwidth. Hence, to differentiate between two vehicles close to each other, the transmitted signal needs to have high bandwidth. The maximum range of the radar is directly proportional to the chirp time (Tchirp), which means that the higher the transmission frequency (lower chirp time), the lower the radar range.

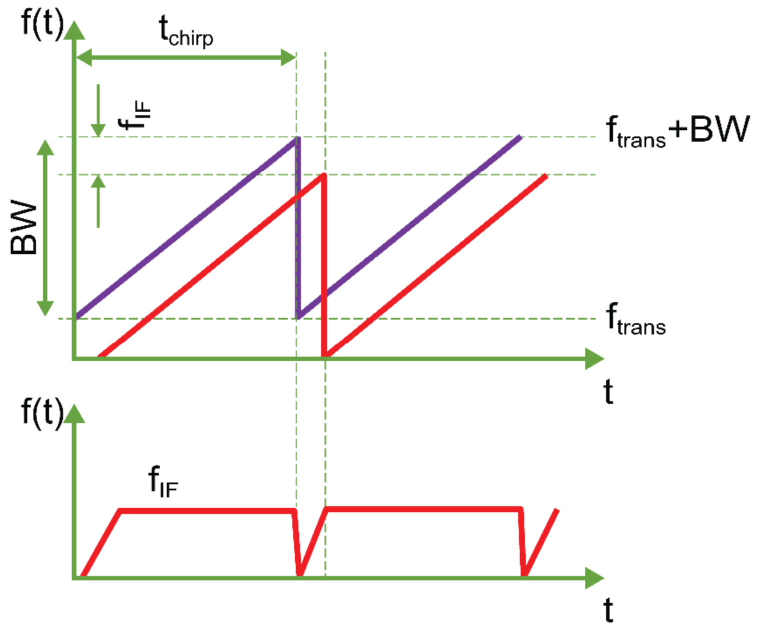


Figure 5.1. RADAR signal transmission and received echo.

This study aims to design and verify a baseband signal processor for an automotive radar with chirp time (Tchirp) = 7.33 µs, sample rate, sweep bandwidth = 150 MHz and overall radar bandwidth (center frequency) equal to 77 GHz. The baseband signal processor was implemented in MATLAB and Simulink (Mathworks Inc.) and the produced HDL code was optimized by applying multiple HLS optimization techniques, such as distributed pipelining and resource sharing.

The end-to-end radar model, which incorporates the baseband processor and a test environment for the processor, is presented in Figure 5.2. The model incorporates an FMCW RADAR, a transmission network, a free space channel, and a receiver network. As displayed in the same figure, the model can be quickly scaled for any number of targets, transmission channel, etc. Applying this model, the design was validated by modeling three separate targets (objects or vehicles to be detected by the RADAR) : a car, a truck, and a pedestrian. A transmitter with an absolute gain of 36 and an isotropic antenna with four elements was utilized for the transmission. Furthermore, a narrowband receiver that uses four isotropic antennas was implemented, accompanied by a receiver pre-amplifier with an absolute gain of 42. Furthermore, the baseband processing system receives the transmitted signal as input and uses a mixer to compute the beat frequency for range and Doppler processing. Simulink's vast library sets and inbuilt functions for multiple objects make the verification environment highly intuitive and simple to model. For this implementation a 2048 point FFT was used along with phase shift beam former. The library objects used are available in phased array system toolbox of MATLAB. Details are mentioned in the Appendix.

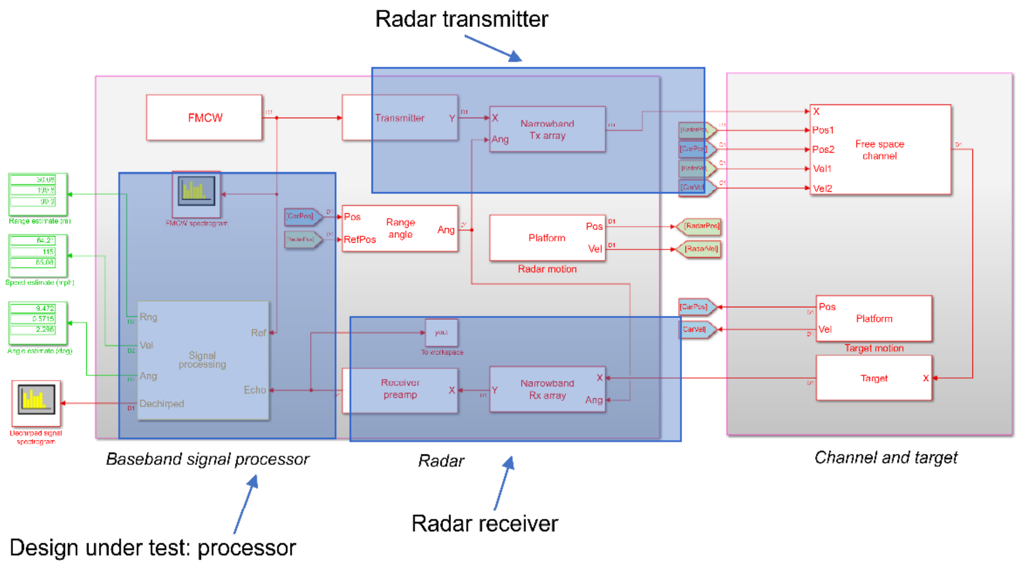


Figure 5.2. End-to-end automotive RADAR system model.

After completing the functional verification tests, MATLAB HDL workflow was employed to produce the Verilog implementation for the receiver system [5.14]. The produced Verilog code was implemented on Kintex-7 FPGA and checked for functionality on real hardware using MATLAB HDL Verifier's FPGA-in-the-loop feature [5.6].The design and verification flow with HLS helped reduce the design cycle time and guaranteed a real-life stimulus to the design for verifying tests at a higher level of abstraction; this made it more exhaustive to cover design functionality. Moreover, we applied various HLS directives such as loop unrolling and resource sharing to optimize the target FPGA implementation results. FPGA-in-the-loop provided superior verification results compared to previous works, and it quickened the verification tests. The FPGA implementation ran faster than the pure simulation in MATLAB and the Verilog HDL simulation that runs on tools such as xSim, ModelSim, or NCSim.

One of the biggest benefits of using this design flow is its flexibility, wherein the designer can experiment with varied architectures for the processor in a common simulation environment before code generation. Once results, such as quantization error and synthesis reports, are within the required limits as per product requirement specifications, code generation is a push-button flow.

5.4 Model simulation and verification results

An end-to-end system model consists of the radar transmitter, antenna, communication channel, and receiver (design under test). Such a system was simulated with two test targets at randomly determined distances and velocities: a car at a distance of 30 m and a truck at 50 m. The car moved towards the origin (transmitter radar) at a speed of 30 km/h, while the truck travelled away from the origin at a speed of 30 km/h. The FMCW spectrum of the transmitted radar signals is presented in Figure 5.3. As shown in the Figure, the signal's bandwidth is 150 MHz (±75 MHz).

The range and doppler map, which are plotted based on the received echo signal, are presented in Figure 5.4. The distance (horizontal) and velocity (vertical) peaks at the points of interest can be clearly observed in the figure. Moreover, the precise distances calculated by the signal processing algorithm running on MATLAB are 30.12 (30 actual) and 50.44 (50 actual) m, and further, the correspondingly calculated velocities are 30.1 km/h (30 kmph actual) and −29.8 km/h (−30 kmph actual) for the car and the truck, respectively. These values are very close to the actual theoretical values for both targets as per the modeling environment (within 1 percent error). The range, velocity, and angle measured by design under test against the actual values (based on target vehicles) for different verification scenarios are listed in Table 5.2. As can be seen from the Table 5.2, the error observed is less than 1 percent for all cases.

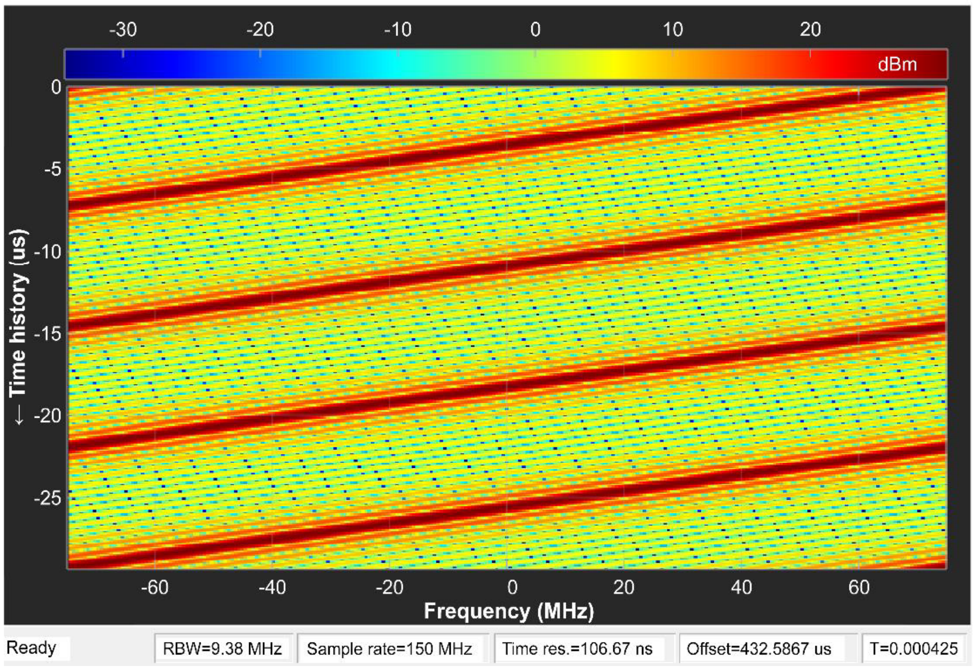


Figure 5.3. FMCW transmission spectrum for automotive RADAR echo.

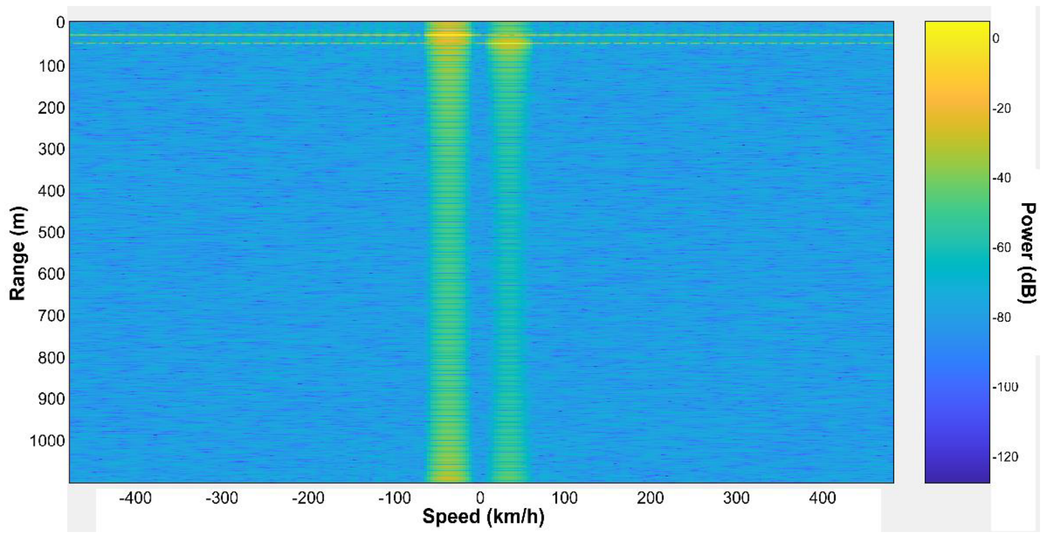


Figure 5.4. Range and Doppler map based on echo received from the target

Table 5.2. Verification test results with modeling of multiple targets

Test Target	Range (Measured/Actual)(m)	Velocity (Measured/Actual) (m/s)	Angle (Measured/Actual) (Degrees)
Pedestrian cross-section	10.67/10.65	1.13/1.10	24.2/24.2
Car cross-section	18.96/18.96	−0.04/−0.07	3.7/3.8
Truck cross-section	42.13/42.07	−0.92/−0.95	−6.8/−6.9

5.5 HLS optimization results for RADAR signal processor

Once the verification using system-level stimuli from the Simulink model was performed, Verilog RTL code was produced for the RADAR processor using HDL coder workflow [5.14]. The RTL code produced was optimized for the area and speed of operation using HLS directives of resource sharing and pipelining [5.15, 5.16]. The scheme of applying these directives is depicted in Figure 5.5. Additionally, we utilized application-specific bit widths as described in Chapter 4 to optimize the design. The RTL produced using the MATLAB HDL coder was implemented on Xilinx Kintex-7 (XC 7K 480T) FPGA and verified for functionality using

MATLAB HDL Verifier's FPGA-in-the-loop methodology. The post-implementation reports from Xilinx Vivado targeting a Kintex-7 FPGA device are summarized in Table 5.3. Table 5.3 also shows the implementation results for the processor when none of the optimization techniques were used i.e it corresponds to the baseline (non-optimal) implementation.

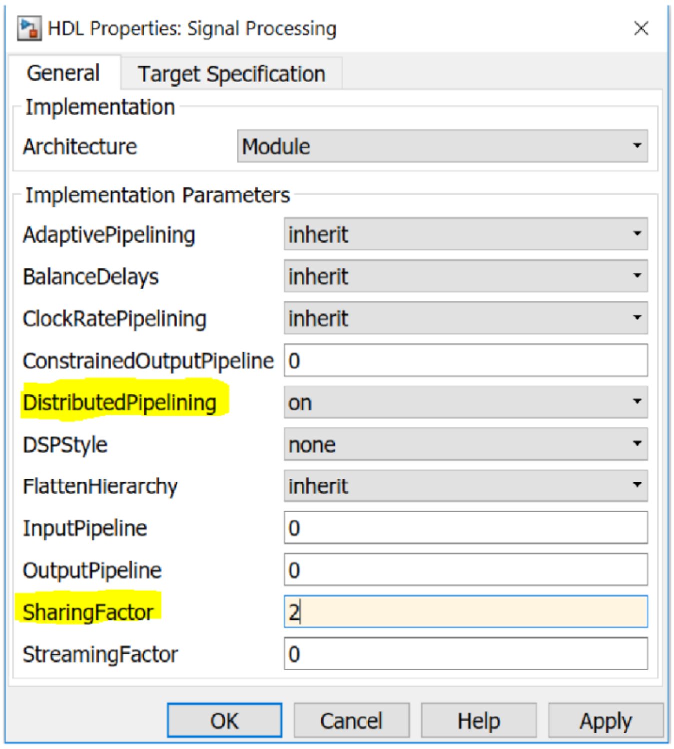
Table 5.3. Synthesis reports targeting Xilinx Kintex-7 FPGA: RADAR processor

Resource	Proposed design	Benchmark design (Default bit widths)	Available Resources
Slice registers	69543	114745	597200
Slice LUTs	51780	88026	298600
Occupied slices	12342	22215	74650
RAMB36E1/FIFO36E1s	367	512	955

Critical Path: 2.43 ns, Operation Freq = 412 MHz, Total On-chip Power = 0.378 W

Table 5.3 clearly shows the results obtained from synthesizing the RADAR signal processor for the target FPGA. It also shows the results of proposed implementation are superior than the pessimistic bit width, default HLS model. The results demonstrated in Table 5.3 have been attained by using 3 optimization techniques:

1) Resource sharing (HLS directive) : As presented in Figure 5.5, a sharing factor of 2 was implemented, which means that a single resource in the design would be shared between two data paths, resulting in lesser resource usage in the design [5.15].


HDL Properties: Signal Processing

General | Target Specification

Implementation

Architecture	Module

Implementation Parameters

AdaptivePipelining	inherit
BalanceDelays	inherit
ClockRatePipelining	inherit
ConstrainedOutputPipeline	0
DistributedPipelining	on
DSPStyle	none
FlattenHierarchy	inherit
InputPipeline	0
OutputPipeline	0
SharingFactor	2
StreamingFactor	0

Figure 5.5. HLS directives of resource sharing and distributed pipelining.

The rationale behind the choice of sharing factor of 2 was because of the fact that a single FFT block was shared by 2 paths in the design – one for range and one for doppler processing.

2) Pipelining (HLS directive): As also presented in Figure 5.5, we enable distributed pipelining for the processor block, which means that all the slower data rate paths would be pipelined automatically, leading to better runtime throughput and performance [5.16]. Moreover, this directive enables the tool to re-distribute pipeline registers to different places on the same combinational path without affecting the functionality. This concept is illustrated in Figure 2.2

3) Application-specific bit widths for intermediate data notes: Rather than using inherited data types for internal nodes (which is by default a 32-bit integer in MATLAB), we choose the intelligent selection of datatypes based on the input stimulus. For example, if the input is the sine or the cosine of a number, its value can lie only between –1 and 1. Therefore, even with an accuracy of up to three decimal points, such as 0.835, we need only two bits for the integer part (–1, 0, or 1) and up to 10 bits for the fraction part (0 to 1024). Hence, we can represent the complete value in 12 bits instead of 32 bits, heading to the design's optimized area usage on the target FPGA. This design is explained in much detail in Chapter 4.

5.6 Results comparison with literature

When the HLS was performed, the design prototype on the target FPGA ran faster, and the synthesis results were optimal in terms of resource usage. These results are associated with the HLS directives, and the bit width optimization methodology applied, as explained in Section 5.5. Table 5.4 compares the resource utilization for the proposed implementation against a related implementation proposed by Sithara et al. with same specifications [5.11]. Even though Sithara et al. used a different wave generation technique (Step Frequency Continuous Wave), the baseband processor specifications are same. Table 5.4 clearly suggests that the proposed implementation uses lesser resources on Kintex 7 as compared to the implementation proposed by Sithara et al.

Table 5.4. Synthesis reports for Kintex7: comparison with the literature

Resource	Proposed design	Sithara et al. [5.11]
Slice registers (%)	11.65	14
Slice LUTs (%)	17.34	19
Occupied slices (%)	16.53	25
RAMB36E1/FIFO36E1s Utilization (%)	38.43	50
DSP48E1s Utilization (%)	16.46	30

A similar implementation from Suleymanov et al. was targeted for Virtex 6 FPGA [5.12]. To have a direct and fair comparison with their implementation, the generated RTL from the optimized high level model was also implemented on Virtex 6 device (XC6VLX240T). Table 5.5 shows the comparison results.

Table 5.5. Synthesis reports for Virtex6: comparison with the literature

Resource	Proposed design	Suleymanov [5.12]
Logic slices	7438 (19.8%)	12158 (32.3%)
LUTs	16324 (6.8%)	30184 (12.5%)
BRAM 36 kBit	56 (13.5%)	70 (16.8%)
DSP48 E1	42 (5.4%)	57 (7.4%)

As is clearly evident from the table 5.5, the proposed implementation is better than that presented by Suleymanov et al. as well. In the literature, some functional results for RADAR signal processor are also presented. These results essentially represent the error in calculation for range, velocity and elevation angle for the target. Such results are available in literature from an implementation proposed by Lin et al. [5.7] and Luthra et al.[5.17]. Table 5.6 shows a complete comparison of the errors in the calculations of the range, velocity, and angle of the target object for our proposed implementation against those presented by Lin et al. and Luthra et al. [5.17].

Table 5.6. Calculation errors: comparison with the literature

Parameter	Proposed design	Lin et al. [5.7]	Luthra et al. [5.17]
Range error (Test1–10.65 m) (m)	0.02	0.03	0.34
Range error (Test2–18.96 m) (m)	0.00	0.05	0.38
Range error (Test3–42.07 m) (m)	0.06	0.1	0.37
Velocity error (Test 1–1.10 m/s) (m/s)	0.03	0.05	2.69
Velocity error (Test 2–0.07 m/s) (m/s)	−0.03	−0.05	NA
Velocity error (Test 3–0.95 m/s) (m/s)	−0.03	−0.03	NA
Angle Error (Test 1–24.2°) (Degrees)	0	0.1	NA
Angle error (Test 2–3.8°) (Degrees)	0.1	0	NA
Angle error (Test 3−6.9°) (Degrees)	0.1	0.4	NA

Tables 5.5 and 5.6 suggest that our proposed implementation is more optimal as compared to other implementations in literature and also has better functional accuracy.

5.7 Concluding remarks

We started this chapter with an introduction to RADAR processors. Their principle of operation, classifications and some of the prior works in the same area were also presented. A RADAR signal processor for automotive ADAS applications was chosen as a test application to apply the learnings from chapter 3 and 4.

An HLS-assisted design and verification framework for automotive RADAR processors was presented. The HLS directives (distributed pipelining and resource sharing) were used to optimize the generated RTL, and consequently, the FPGA implementation.

Additionally, we utilized application-specific bit widths for the intermediate nodes, a novel methodology developed for the HLS as described in chapter 4 to optimize the implementation further. The design was validated using an environment modeled in Simulink using FPGA-in-the-loop feature of MATLAB. The results show that the proposed implementation calculates the distance, speed, and angle of the targets relative to the RADAR position, more accurately than other implementations presented in previous research. Moreover, the results are identical to those achieved by post-silicon application teams in the field testing for the same design.

Synthesis and implementation reports for both Virtex-6 and Kintex-7 FPGAs clearly imply that the proposed implementation is better in terms of resource usage for the same FPGA target as described in other studies. The total time to implement the FPGA design is also considerably shorter in the proposed implementation, as it is based on HLS and does not use a conventional hand-coding approach for RTL. Moreover, the framework yields flexibility for the designer to iterate between multiple implementation options for the same architectural specifications for no additional cost or effort, which is a typical advantage of HLS flow.