2011 21st International Conference on Field Programmable Logic and Applications

Heterogeneous Platform for Stream Based Applications on FPGAs

Jan Kloub, Tomáš Mazanec and Antonín Heřmánek
Institute of Information Theory and Automation of ASCR
Pod Vodárenskou věží 4
182 08 Praha 8, Czech Republic
Telephone: (+420) 26605 2472
Fax: (+420) 26605 2511
Email: {kloub, mazanec, hermanek}@utia.cas.cz

***Abstract*—The complexity of embedded systems is ever increasing, to support wide range of application domains. Dedicated hardware provides improved performance but with limited resources reusability and application scalability. Using higher abstraction methods for hardware development shortens the time to market. The concept of *hardware objects* (HWO) allows for more efficient hardware resource reuse and scalability of the target application. A single task of the application can be mapped on an appropriate type of hardware structure. A single hardware structure can be shared in time by several application tasks; multiple instances of the same HW structure can be used for concurrent computation. Uniform hardware object structure shortens design time and simplifies design of target system independently on application domain. The system could contain a heterogeneous set of HWO instances to satisfy all application domains.**

I. Introduction

State-of-the art wearable consumer devices, such as smart phones or tablets, support wide range of application domains e.g. audio and video applications, satellite navigation and digital data communications.

Design of such devices represents a three dimensional problem where a trade off between performance, power consumption and design cost has to be done. There is no universal solution to satisfy all aspects of the problem. Customized hardware solution brings performance, low power consumption but limited reusability; general purpose solution (CPU) brings high reusability but lower performance. As the computation time on CPU is longer than on customized hardware, it consumes more power.

One of possibilities to improve the reusability of the customized hardware structures is to extract reusable computation constructs from each application domain. Constructs are fully customized and additional hardware allows for their reuse in different applications. For example, the 2-D convolution core can be used in many image processing applications [5] (filtering, edge detection, weak classifier etc.), floating point arithmetic unit can be used for vector operations [3], complex arithmetic unit for signal transformations (FFT, Discrete Wavelet Transform) or in digital communication (ADSL, GSM, WiFi etc). Similar techniques are used for design of application specific (vector) processors [9].

This paper presents system design using the hardware object (HWO) concept. Each active part of application is mapped on the hardware structure which provides appropriate object methods. Let us note that the hardware resources can be heterogenous while the structure is uniform.

A uniform HWO structure, described in [5], provides consistent interface to the system which allows for more efficient implementation of target application software part and hardware functional compatibility. Several application tasks can be computed concurrently if more HWO instances are available or HWO can be shared in time if (partial) dynamic reconfiguration is used [1]. Such system/architecture corresponds to the concept of Software Defined Radio (SDR) [7] and brings the possibility to develop new devices such as cognitive radio [2] or smart mobile terminals.

The paper is organized as follow: Background and related work is described in the section II, implementation of DAB receiver physical layer is described in the section III and implementation results are summarized in the section IV.

II. Background and Related Work

To extract a set of common operations constructs for OFDM communication systems, profiling of DVB-T2, DAB and WiFi/WiMax applications were performed. Complex computing elements (CCEs) was implemented to support arithmetic computation in complex domain for signal processing algorithms used in digital communication systems. The structure of CCE is based on a previous work on basic computing element (BCE) [3] and graphic computing element (GCE) [5].

A simplified structure of GCE, BCE or CCE is presented in Figure 1. The data flow unit (DFU) establishes a data chain composed of input memory(ies), functional unit(s) and output memory. When operation is started an address generators of input and output memories are driven by the DFU. Linear addressing and circular buffer addressing are supported. All functional units are fully pipelined, to achieve high data throughput for vector operations.

Interface and control logic (simple CPU) are uniform for all computing elements in the system to allow application tasks running independently on the given implementation of

978-0-7695-4529-5/11 $26.00 © 2011 IEEE
DOI 10.1109/FPL.2011.80

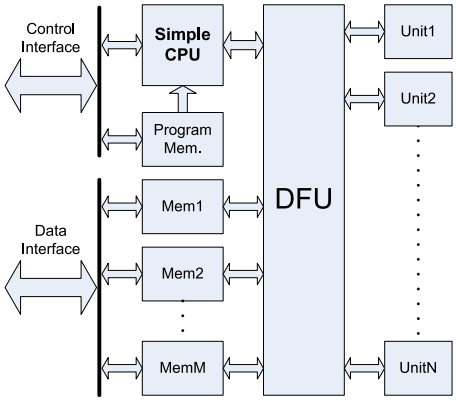

Fig. 1. The structure of computing elements (BCE, GCE, CCE)

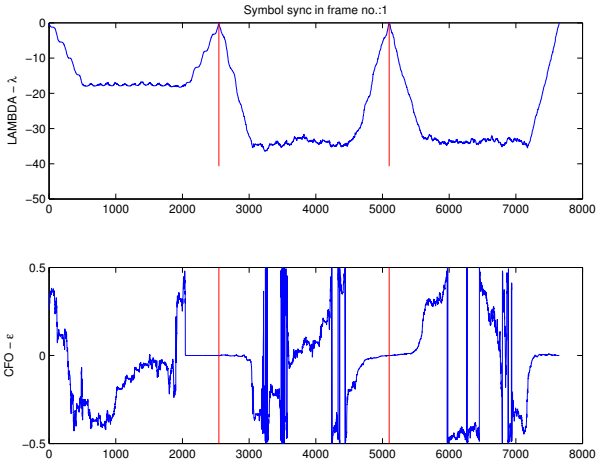

Fig. 2. Joint time and frequency offset estimator outputs for symbol synchronization and frequency offset (SNR $= 50\,\mathrm{dB}$, $\epsilon \rightarrow 0$), red is sync on t_{max}

DFU and functional units. More complex tasks have to be decomposed to the elementary operations supported by the DFU with sequencing controlled by the CPU firmware.

The resulting computing element types are used as HWO instances for appropriate application domain.

III. DAB Implementation

The implementation of physical layer of *Digital Audio Broadcast, DAB,* receiving system that corresponds to DAB standard [4] was chosen as a case study for implementation and testing of the complex computing element (CCE).

The CCE consists of the following functional units:

- Complex Adder
- Complex Multiplier
- Square Root
- Comparator
- FFT Core[1]

The real and imaginary parts of the complex number have 16-bit floating point representation. The 16-bit floating point arithmetic is a compromise between amount of hardware resources and precision. Data have 32-bit representation where upper 16 bits are for real and lower 16 bits for imaginary part of complex number. Implemented CCE has four memory banks. The length of the memory banks was chosen with respect to the longest DAB OFDM symbol size (2552 samples) to 4096.

The following parts of the DAB receiver physical layer have been implemented:

- Null symbol detection, frame synchronization and partitioning
- symbol synchronization based on guard interval (GI)
- estimation of fractional and integer carrier frequency offset (CFO)
- compensation of estimated CFO
- symbols partitioning
- guard interval removal
- symbol demodulation by FFT
- channel estimation and compensation
- frequency deinterleaving

All steps above, except the frequency deinterleaving, were implemented using CCE structure. Frequency deinterleaving was implemented in software on master processor (MicroBlaze), and therefore it is not part of this paper.

Let us show, as a typical representative of the algorithms used in our receiver implementation, the symbol synchronization algorithm. Its mathematical description and implementation will be presented in the following text.

Joint time and frequency offset estimation [6], which is commonly used because of its low complexity, has been used in the DAB implementation. This algorithm evaluates a metric $\lambda(t)$ (1) composed of two correlation elements $\gamma(t)$ and $\kappa(t)$, where the latter is exploited to determine the Null symbol occurrence in the DAB frame. The metric is described as follows:

$$\lambda(t) = |\gamma(t)| + \kappa(t) \tag{1}$$

where

$$\gamma(t) = \sum_{k=0}^{N_g-1} s(t+k)s^*(t+k+N) \tag{2}$$

and

$$\kappa(t) = -\frac{1}{2}\sum_{k=0}^{N_g-1} \left(|s(t+k)|^2 + |s(t+k+N)|^2\right) \tag{3}$$

where $s(t)$ is the input signal, t is discrete time index, $(.)^*$ is complex conjugation, N is the OFDM symbol size and N_g is the guard interval size.

The estimator output, $\lambda(t)$, peaks at the beginning of the OFDM symbol as can be seen in Figure 2. Fractional CFO

[1]For target application performance increase the programmable Xilinx FFT core were used instead of using standard functional units.

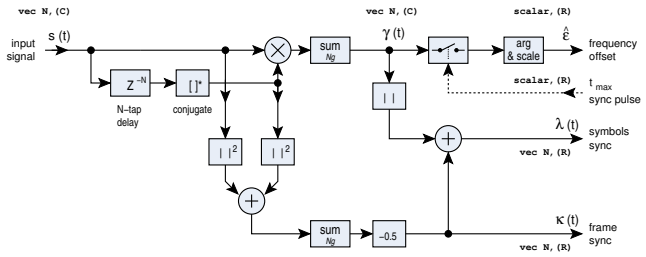

Fig. 3. Diagram of the joint time and frequency estimator

Task	MUL	ADD	CMP	SQRT	FFT	MEM
Freq. Est.	5	6	1	1	0	3056
Integer CFO	3	1	1	1	0	4101
FFT	0	0	0	0	1	0
Chan. Est.	1	0	0	0	0	4096
D-QPSK	2	0	0	0	0	6144
Summary	11	7	2	2	1	17397
CCE	1	1	1	1	1	16384

TABLE I
HARDWARE RESOURCES ESTIMATION FOR FULLY PIPELINED IMPLEMENTATION WITH COMPARISON OF CCE CONFIGURATION

can be determined by the equation (4) from the $\gamma(t)$ at the peak positions.

$$\hat{\epsilon} = \frac{1}{2\pi} \arg\left(\gamma(t_{max})\right) \tag{4}$$

The joint time and frequency estimator can be described with block diagram depicted in Figure 3. Each block of the diagram is implemented by one or more consecutive vector operations.

For example, the sum_{N_g} block can be implemented as sliding window summation described by equation (5).

$$sum_{N_g}(t) = sum_{N_g}(t-1) + s(t) - s(t-N_g) \tag{5}$$

Note: Such a solution can lead to numerical instability due to round-off errors namely when the floating point number representation is used. In our case, we use relatively short vector lengths and regular reinitialization.

The CCE solves the equation (5) by four consecutive vector operations as described by the following pseudo-code:

```
// Initial summation
D[1] <= SUM(A[1..Ng]);

// Subtract part
B[1..M-Ng] <= CUMSUM(A[1..M-Ng]);

// Additional part + Initial sumation
C[1..M-Ng] <= CUMSUM(A[1+Ng..M]) + D[1];

// Update sliding sum
A[1..M-Ng] <= SUB(C[1..M-Ng], B[1..M-Ng]);
```

where `M` is the input data length and `A, B, C, D` are the CCE memory banks. The first instruction corresponds to initial value of the sliding window, the second instruction computes partial decrements of the initial window value and the third computes partial increments plus the initial value. The last instruction computes the result of sliding window summation.

All the rest of receiver algorithms are implemented in the same way as shown in the sliding summation example.

The master processor stores the input data into the CCE memory banks and calls the CCE methods to process the data. Each of the method calls correspond to appropriate functional block of the DAB receiver.

The presented solution uses CCE as HWO instances to implement the following functional blocks: frequency estimator, integer CFO detection, channel estimation & compensation, OFDM symbol and D-QPSK demodulation.

The application starts as follows: first, the null symbol detection and frame synchronization is performed using the equation (3). This computation is performed on all input data until the null symbol is detected. Next, the symbol time and frequency offset estimation, as presented in equation (1), is performed, where the computation is evaluated only at the regions close to the symbol boundaries. The computation time and data communications are significantly shortened and the CCE can be used for another purpose (another HWO instance) until next input data are ready.

The integer CFO can be modified during the runtime in the same way. Wide frequency interval is scanned at the start up and only short interval afterwards because frequency variation is limited in real application.

To be able to evaluate the reuse ration of our approach, the estimation of resources for the implementation using a common design methodology has been performed. Physical layer of receivers is usually implemented as fully pipelined design with either no or poor possibility of functional unit reuse. The estimation of the required HW resources is evaluated as the number of functional units (complex adder, multiplier etc.), required for pipelined implementation of given algorithm. The resource estimation for pipelined implementation and comparison with CCE is summarized in Table I.

The pipelined design may run at the data sample rate frequency or on higher frequency with clock enabling, where the latter is more common because the data rate frequency of communication systems is usually significantly lower than clock rates of the current processor-based platforms. As a consequence functional units are not fully utilized. Using HWO concept, the high reusability of functional units is allowed because the units can run at maximal clock rate, while being shared in time between tasks.

IV. RESULTS

For evaluation of the DAB receiver physical layer implementation, the Xilinx development kit "XtremeDSP Development Platform Spartan-3A DSP 3400A Edition" [8] was used. The system consists of the MicroBlaze master processor, two instances of the CCE connected through the PLB bus running at the 62.5 MHz clock rate. The data transfers are provided by Xilinx Central DMA controller.

FPGA Resource	CCE	CCE-FFT
Slices	3279/23872 (13.7%)	2166/23872 (9.1%)
FFs	2350/47744 (4.9%)	2063/47744 (4.3%)
LUTs	4461/47744 (9.3%)	2359/47744 (4.9%)
BRAMs	36/126 (28.6%)	17/126 (13.5%)
DSP48A	0/18 (0%)	5/18 (27.8%)

TABLE II
COMPLEX COMPUTATION ELEMENTS HARDWARE RESOURCES UTILIZATION SUMMARY (XILINX SPARTAN-3A DSP 3SD3400AFG676-4)

Implementation	Computation Time [μs]
Sequential	55706.13
Parallel	23757.89
DAB Constraint	96000.00

TABLE III
ESTIMATED DAB FRAME COMPUTATION TIMES FOR SEQUENTIAL AND PARALLEL IMPLEMENTATION SCENARIOS

The first CCE instance contains standard functional units (adder, multiplier, etc.) and the second instance (CCE-FFT) programmable Xilinx FFT core only. The 16-bit fixed point FFT core is used where the floating to fixed point transformation is performed directly in the hardware. The CCE-FFT instance has been used to increase the performance of the application. The CCEs hardware resource utilization is summarized in Table II.

Physical layer of DAB receiver is composed of several tasks (HWO methods) which have to be performed in each data frame. The Table IV shows task computation times in CCEs hardware. It is possible to perform some tasks concurrently, for example, when the start of the frame is known, the frequency estimator can run independently of the rest of tasks because the positions of the symbols are known then.

We have evaluated the following two basic implementation scenarios with respect to task parallelization:

- Sequential – All tasks are performed in successive sequence for each symbol.
- Parallel – Frequency estimator and symbol demodulation (FFT) run in parallel to the other tasks. The symbol demodulation is done in parallel for all symbols in the frame.

Estimated computation times per frame of these scenarios are presented in Table III. The presented computation times are without any communication overhead. The DAB frame constraint is 96 ms, so there is space for some overheads in all presented cases.

The sequential version has been implemented for performance evaluation. To measure the computation time precisely, the Xilinx timer has been used. Measured frame computation time is 77771 μs and data transfer time is 113484 μs (@62.6 MHz). The measured time includes software part of DAB application running on the master processor. A significant overhead of the application represents the data transfers. One of the main reasons is the use of only one PLB data bus, which is shared between CCE instances, master processor and DMA controller. Dedicated data paths or use of multiple PLB will solve the problem and will be done in future work.

Task	Tasks per frame	Exec Time [μs]
Freq. Est.	77	308.54
Integer CFO	1	740.24
FFT	76	284.10
Chan. Est.	1	101.70
D-QPSK & Chan. Compensation	75	126.88

TABLE IV
DAB TASKS PROFILING (@62.5MHZ; SCANNED INTERVAL = GI = 504 SAMPLES, INTEGER CFO = 2)

Very high computation construct reuse was achieved in the presented implementation (see Table I for details), e.g. 11 times in case of multiplier and 7 times in case of adder.

V. CONCLUSION

This paper presented implementation of the Digital Audio Broadcast receiver physical layer as the case study of the hardware object concept. The hardware objects are directly mapped on computing elements which have uniform structure. The uniform structure of computing elements reduces design time of a new class of hardware objects. An essential part of computing element can be inherited (as in software object implementation) and only small modification can be done to support new object methods or its implementation or data representation. The implemented CCE has been developed for long vector operation used in communication applications. High degree of hardware resources reuse with satisfactory performance was achieved in the case study.

ACKNOWLEDGMENT

This work was supported by the SCALOPES project; project number: Artemis JU 100029, MSMT 7H09005.

REFERENCES

[1] Grüll F. Meier N. Beyer A. Kebschull U. Abel, N. Parallel hardware objects for dynamically partial reconfiguration. pages 563–566, Heidelberg, 2008.
[2] P. Athanas. SDRF cognitive radio definitions,working document SDRF-06-R-0011-V1.0.0, November 2007.
[3] Kadlec J. Bartosinski R. Kohout L. Daněk, M. Increasing the level of abstraction in FPGA-based designs. pages 5–10, Heidelberg, 2008.
[4] European Telecommunications Standards Institute, 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France. *ETSI EN 300 401 V1.4.1: Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers*, publication edition, June 2006. Reference REN/JTC-DAB-36.
[5] Honzík P. Daněk M. Kloub, J. Reconfigurable hardware objects for image processing on FPGAs. page 121, Vienna, Austria, Apr 2010.
[6] G. Li and G. L. Stüber (Eds.). *Orthogonal frequency division multiplexing for wireless communications*. Springer Science+Business Media, Inc., 1st edition, 2006.
[7] W.H.W. Tuttlebee. *Software Defined Radio, Baseband technology for 3G Handsets and Basestations*. John Wiley & Sons, Ltd, 1st edition, 2004.
[8] Xilinx Technical Repository. *XtremeDSP Development Platform: Platform: Spartan-3A DSP 3400A Edition, User Guide*, September 2007.
[9] Jason Yu, Christopher Eagleston, Christopher Han-Yu Chou, Maxime Perreault, and Guy Lemieux. Vector processing as a soft processor accelerator. *ACM Trans. Reconfigurable Technol. Syst.*, 2:12:1–12:34, June 2009.