

APPLICATION CENTRIC PROCESSING

Dr. Bob Madahar,
Mr. Brian Ford,
Mr. Andrew Mackey

BAE SYSTEMS Advanced Technology Centre,
Gt. Baddow, UK

1 INTRODUCTION

Increasing complexity in future sensor based systems, places exacting demands on development of embedded signal processing in these systems. The design flow methodologies used to address future systems need to support critical business drivers, such as reduction in life cycle costs, affordability and obsolescence management, and crucially the verification and validation activities, whilst fully exploiting the technical potential of current and future embedded technology.

This paper presents the outcome of quantitative assessments of three design flow methodologies that have been chosen to give good example coverage of methodologies from those in current use through to those that might be expected to become the norm in 3 to 5 years.

To undertake the assessment, a representative embedded processing design challenge was identified. The target implementation was a Field Programmable Gate Array (FPGA) and the design, already available in a common Hardware Description Language (HDL), was re-implemented for evaluation, refinement and trade-off studies using system level design tools and techniques.

2 REQUIREMENTS, TOOLS & METHODS

2.1 Requirements for TEP Design Methods

Early work **1 identified the** critical requirements and business drivers for design methodologies for TEP for EMRS systems. The increasing use of Field Programmable Gate Array (FPGA) technology to provide re-configurable hardware has also been identified as a key supporting technology **2**.

The top level critical business driver requirements for future design methods were captured **1** as:

- Reduced life cycle costs and time.
- Obsolescence management.
- Certification (Validation and Verification).
- Affordability and availability.

System level design languages are key to achieving these requirements by providing:

- Abstraction of design to a high level.
- Design source re-use (Code/Schematic).
- Enable hardware and software co-designs.

However issues arising from system level design languages for targeting transducer embedded processing devices such as FPGA, include:

- Maintaining design efficiency.
- Controlling latency to area trade-offs.
- Controlling latency to power trade-offs.
- Leveraging the inherent parallelism in FPGAs.
- Partitioning design over multiple FPGAs.

It is these last five points that were of particular interest in the work described in this paper.

2.2 Languages and Tools

The requirements capture exercise 1 undertaken during the EMRS DTC project also identified a range of candidate tools and languages. These are summarised briefly here as an introduction as to how suitable candidate 'Design Flows' were chosen for the quantitative assessment.

2.2.1 Hardware Design Languages

Hardware design languages are traditionally used for the creation of designs suitable for FPGA or ASIC implementation. There are several languages in this class, two of the most common are:-

- VHDL
- Verilog

VHDL is an IEEE standard language 3 for hardware description at various levels of abstraction. The language has the flexibility to describe very high level behaviour as well as low level gate behaviour. Due to the hardware abstraction available with VHDL the language is widely used, both as the primary design language and as an intermediate language emitted from high-level system design tools.

The Verilog language was originally proprietary, however has since become an open IEEE standard language 4 for hardware description at various levels of abstraction.

2.2.2 System Level Languages for FPGA & ASIC

This class of language is tailored to support FPGA or ASIC but maintains a syntax similar to that used in software engineering. Two examples in this class are:

- Precision C
- Handle-C

Precision C is an emerging tool from Mentor Graphics 5. The tool offers pure C/C++ synthesis to target FPGA or ASIC.

Handel-C is a combined language and development environment from Celoxica 5 that is sufficiently similar to ANSI C to allow software engineers to progress rapidly with a design for FPGA, without extensive knowledge of the underlying hardware. The language has many extensions to ANSI C to better exploit the underlying hardware of an FPGA, including facilities to support the parallelism available in FPGAs.

2.2.3 Target Independent System Languages

In this class are the group of tools and languages that purport to offer truly target independent system level design (rather than designs for assumed targets such as ASICs or FPGA devices). These are:

- SystemC
- GEDAE
- Matlab/Simulink

SystemC is an open source language promoted by the Open SystemC Initiative (OSCI) 7 to provide a design and verification language. The language provides a hardware-orientated class library implemented in C++, allowing SystemC designs to be modelled in a C/C++ environment. The OSCI includes several

leading EDA vendors, to provide tools to directly target hardware or other HDLs.

GEDAE is a data flow tool from Blue Horizon Development Software 8 and provides a graphical design entry and simulation environment. The tool provides automatic code generation to support a number of DSP and RISC targets and ultimately FPGA¹.

Matlab is well established mathematical modelling language from Mathworks 9 and together with **Simulink** (schematic block front end) is commonly used for capture, simulation and modelling of TEP systems. The tool provides a high-level programming language with advanced graphics and visualisation.

At least two FPGA vendors offer tools to allow DSP functionality for FPGA as Matlab/Simulink blocks. Altera 10 offer the ‘DSP builder’ and Xilinx 11 the ‘System generator for DSP’ tools.

3 DESIGN FLOWS

Three design flow methodologies were selected for evaluation. The selected design flow methodologies were chosen to give good example coverage of methodologies from those in current use through to those which might be expected to become the norm in 2 to 5 years.

3.1 Design Flow 1 - Matlab Model, VHDL Implementation.

This is considered the baseline Design Flow and represents ‘current practice’. The design flow is shown in Figure 1 and consists of a Matlab model of the design and an implementation of the design in VHDL.

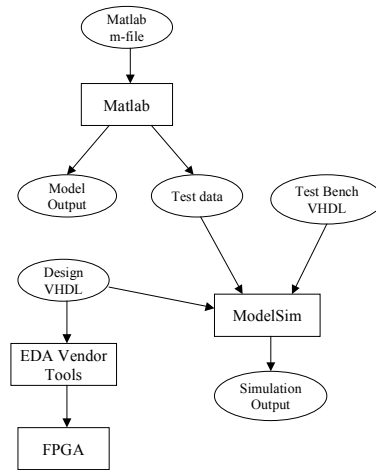


Figure 1: Structure of Design Flow 1

3.2 Design Flow 2 - GEDAE

This design flow was chosen to represent the ‘state of the art’, and likely to represent current practice for TEP design flows within the next 5 years. The tool provides a very high level of abstraction and design capture is via a hierarchical block diagram form, independent of the final target platform. The structure of the design flow is shown in Figure 2. The graphical and hierarchical nature of the design capture provides a high level of readability (and documentation) for the design. The tool also provides detailed timing analysis when running as a simulation or on the final platform. There also exists the future possibility of support for FPGA. Thus this tool has the potential to provide a very readable and maintainable design.

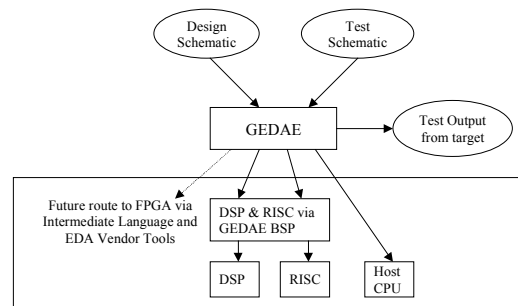


Figure 2: Structure of Design Flow 2

¹ The version of the Tool used in the evaluation does not support FPGA

3.3 Design Flow 3 - Matlab Model, Handel-C Implementation

This design flow was chosen to represent the future baseline (2 to 3 years) practice for TEP design flow. Design Flow 2 provides the ability to exploit FPGA targets and architectures whilst maintaining a sufficient level abstraction from the hardware. The structure of the design flow is shown in Figure 3. The Matlab model from Design Flow 1 is used in this design flow to provide verification of the design.

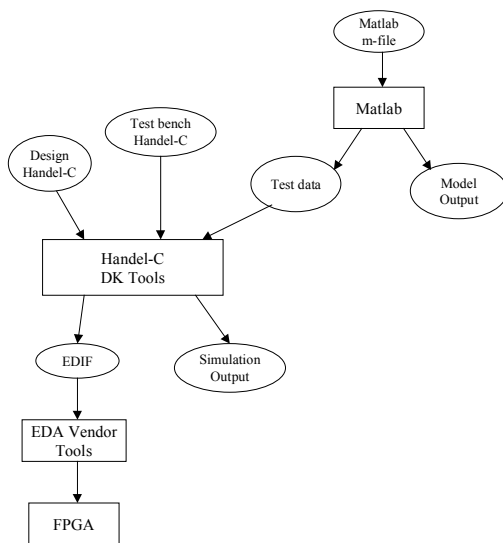


Figure 3: Structure of Design Flow 3

Note that there are three possible intermediate file formats between the Handel-C and EDA vendor tools available within this design flow. The EDIF format was chosen over the other possibilities of VHDL and Verilog, as EDIF is specifically designed as a gate level electronic design interchange format between EDA tools.

4 CANDIDATE DESIGN AND METRICS

4.1 Metrics

For the purposes of undertaking a quantitative assessment of design flows, four metric ‘groups’ were defined to evaluate the methodologies. These groups were defined following the requirements capture exercise 1 and are based on the

‘ilities’ regarded as important for design flow evaluation.

1. Affordability
 - Tool costs
 - Development time
2. Technical capability
 - Processing time (or clock cycles)
 - Device usage
 - Throughput
3. Testability
 - Bit level simulate at I/O
 - Time to implement test harness
4. Maintainability
 - Lines of code

The objective was to capture relevant quantitative metrics for each metric group, however in the case of testability only qualitative measures were possible.

4.2 Design Challenge Example

To provide a suitable TEP design challenge the channelisation (FDM) of a broad band digitised IF/RF signal was chosen. Several candidate architectures are capable of delivering this functionality (e.g. Multiple decimating FIR, FFT, cascaded half band FIR).

The design challenge was implemented using a cascaded decimating FIR Half band filter for the channelisation process. In this arrangement a single half band filter stage is cascaded to give the multiple channels.

4.3 Target Device

The target FPGA selected for the evaluation of the design flows was the Xilinx VirtexII. The particular part is the XC2V6000, speed grade –4 with enhanced multiplier blocks.

5 DESIGN IMPLEMENTATION

This section gives details of the implementation of the design challenge in each of the chosen design flows.

5.1 Design Flow 1

The Matlab simulation is a fixed point arithmetic model of the functionality of the VHDL implementation. The model is

capable of emitting suitable test data for testing the VHDL implementation. The output of the VHDL simulation can then be directly compared with the Matlab model output.

The VHDL implementation was fully optimised for the architecture of a half band FIR stage and utilises the hardware multipliers available in the Xilinx device. The design is pipelined to achieve a high throughput, with no unused or 'waiting' clock cycles.

5.2 Design Flow 2

The GEDAE design was implemented using standard parts from the library. The filter design consists of a FIR block connected to a decimator block, with the filter coefficients set to give a half band filter response. Unlike the Handel-C and VHDL designs this design is implemented in floating point arithmetic. The design allows for many of the half band stages to be cascaded as required, by the 'stages' parameter shown on the top-level diagram of Figure 4.

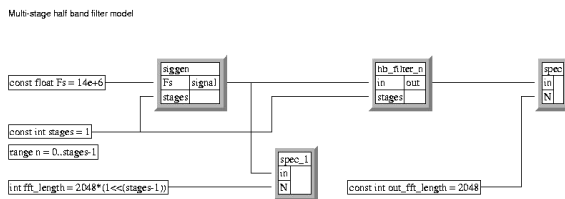


Figure 4 GEDAE half band filter multi-stage model

5.3 Design Flow 3

This design flow uses the Matlab model from Design Flow 1 to provide test data for the Handel-C simulator to use. The Handel-C tool provides a channel interface that supports simple I/O to a data file. The data emitted by the Matlab model was not in a suitable format to use this simple interface and to further investigate the interface possibilities of Handel-C, a small ANSI-C function was written to facilitate reading the input data. The built in channel

I/O was used to record the output from the simulation.

The Handel-C implementation of the half band filter stage has several variants to explore how the structure of the code can effect the performance of the design. These are described in the following subsections.

5.3.1 Handel-C Sequential Design

This is a standard FIR design with filter coefficients to give a half band response. The design was implemented using standard C syntax where possible. The output is calculated by using sequential loops the same as would be used in a standard C implementation. No consideration to the final device was given so any device specific accelerations were not used.

5.3.2 Handel-C Optimised Sequential Design

This design was specifically optimised for the half band architecture. The output is calculated by using sequential loops the same as would be used in a standard C implementation. The design has three sub-variants, (ref (a), (b) and (c)) to investigate specific design decisions.

Variant (a) and (b) of this design are identical apart from the use of the hardware multiplier available on the Xilinx target, used to assess the impact of this multiplier. The third variant (c) uses more intermediate registers to break long combinatorial logic paths to investigate the effect this has on clock speed.

5.3.3 Handel-C Optimised Parallel Design

This is a design specifically optimised for the half band architecture and utilises the 'par' extension to produce a low latency parallel design. In addition the design also makes use of the hardware multipliers available on the Xilinx target.

6 RESULTS

6.1 Technical Capability

The performance and usage characteristics for each of the final design flows are shown in Table 1 for comparison.

| Design Flow | | Clock Speed (MHz) | Latency (Clock cycles) | Device Usage (Slices) |
|-------------|------------------------------|-------------------|------------------------|-----------------------|
| 1 | VHDL | 120 | 10 | 226 |
| 2 | GEDAE | N/A | N/A | N/A |
| 3 | Handel-C, Optimised Parallel | 60 | 2 | 403 |

Table 1 - Speed, latency and area comparisons of Design Flows

Note that version of GEDAE available to this work did not include the capability to target the specific FPGA device. Nevertheless a processing time measurement was taken for the code when running on a PentiumII 447MHz host. The total processing time for one stage of the half band filter has been measured as 29 μ S. The several variants of Design Flow 3 were employed to investigate the effects of code structure on the efficiency of the implementation, are compared in Table 2.

| Design variant | Clock Speed (MHz) | Latency (Clock cycles) | Device Usage (Slices) |
|-----------------------------------|-------------------|------------------------|-----------------------|
| Handel-C Sequential | 38 | 17 | 535 |
| Handel-C Optimised Sequential (a) | 37 | 6 | 415 |
| Handel-C Optimised Sequential (b) | 53 | 6 | 263 |
| Handel-C Optimised Sequential (c) | 90 | 14 | 296 |
| Handel-C Optimised Parallel | 60 | 2 | 403 |

Table 2 - Speed, latency and area comparisons of Design Flow 3 (Handel-C) designs

6.2 Testability

The testability of the three design flows was captured using qualitative assessment. For each of the design flows the testability strength and weaknesses of the flow are presented as follows:

Design Flow 1

- Strengths - Bit accurate Matlab model comparison with VHDL design. Accurate timing simulation possible.
- Weaknesses - Additional tool (ModelSim) required in design flow.

Design Flow 2

- Strengths - Integrated design and test environment. No need for separate high level model. Facilities to identify memory usage and execution time.

- Weaknesses – Proprietary scheduler used to control design execution.

Design Flow 3

- Strengths - Bit accurate Matlab model comparison with Handel-C design.
- Weaknesses – Accurate timing simulation not possible with Handel-C simulator alone.

The Handel-C simulation allowed the design to be functionally simulated and the output obtained verified against the VHDL and Matlab models. However there is no integrated way for verifying an accurate timing simulation within the Handel-C environment. To achieve this the FPGA vendor tools are required. In the Xilinx scenario it is possible to create a VHDL test bench skeleton from the exported Handel-C EDIF for use with ModelSim, but this test bench is independent from the Handel-C simulation.

For the GEDAE implementation the design was simulated within the tool by designing a suitable input signal generator equivalent to that used in the Matlab model. The tool provides a well integrated environment for the simulation of the design.

6.3 Affordability

The development time for the designs was recorded to give an indication of affordability of the design flows. Table 3 gives the development time of the example design associated with each design flow.

| Design Flow | Design Flow Component | Time (days) |
|-------------|------------------------------|-------------|
| 1 | Matlab | 5 |
| | VHDL Initial design | 15 |
| 2 | GEDAE | 5 |
| 3 | Matlab ² | 5 |
| | Handel-C | 5 |
| | Handel-C (each optimisation) | 5 |

Table 3 - Development time with each Design Flow

Both the GEDAE and most Handel-C designs were implemented within one week, including in the GEDAE case the test signals and output display, and for the Handel-C case the interface to read the Matlab input. However for the Handel-C implementations a further week was spent on optimising the designs.

The Handel-C synthesis for the chosen FPGA family gave an indication of the size and maximum delay paths in the generated logic, which was used to optimise the design.

To get more accurate timing information for optimisation, the FPGA vendor tools may be used. For the Xilinx target, the techniques set out in the Celoxica, Timing Analysis application note 12 have been used to optimise the design. Using this route increases the development time as a full place and route of the design is required.

6.4 Maintainability

The number of code lines is a basic measure of ‘maintainability’. Factors such as code readability, parameterisation, commenting and layout are also important factors defining the ‘maintainability’ of the code. Nevertheless for the purposes of presenting

² Matlab model re-used from Design Flow 1.

quantitative data Table 4 shows the results for design flow maintainability.

| Design Flow | Design Flow Component | Lines |
|-------------|-----------------------------|-----------------|
| 1 | VHDL | 110 |
| 2 | GEDAE | N/A |
| 3 | Handel-C Optimised Parallel | 92 ³ |

Table 4 –Lines of code to implement core FIR algorithm

Note that the lines of code reported in Table 4 are for the core code to implement the FIR algorithm. Both the VHDL and Handel-C have additional code not listed in Table 4 that control the core FIR code.

Also note that for the design in GEDAE is split over 5 schematic data flow diagrams. The graphical nature of the design input provides a high degree of readability. The structure of the design and constants used can be set out on the diagram, as **Figure 4** shows.

7 CONCLUSIONS

This paper has focussed on comparing the technical issues between three different design flows, to identify the issues effecting the efficient implementation of an example TEP design.

The results of the assessment show that for Design Flow 3, while it is possible to produce a Handel-C design using much of the same syntax as ANSI-C ignoring hardware considerations, the implemented performance of the design is poor when compared to the VHDL implementation. By considering the hardware, the performance of the design can be improved to approach that of the VHDL design. However by optimising the design, the

³ This design call a macro to select at build time if the xilinx hardware multiplier is used. This macro is not included in the line count and resolves to about 3 lines of code.

readability of the original code is obscured by instructions dealing with the structure of the hardware.

Design Flow 3 (Handel-C) offers a level of readability to software engineers not familiar with traditional hardware HDL's. However this perceived level of understanding must be treated with caution if the source code expressing the design has been optimised for the target hardware. The results from Table 2 show that small differences in source code can have a very significant effect on the efficiency and speed of the final item.

Many FPGA devices have special purpose blocks such as dedicated multipliers to increase performance. The ability of system level design tools to transparently use these features is important to keep source portable.

Design Flow 2 (GEDAE) has proved to be rapid and easy to use, to construct both the test bench and design. This design flow has the advantage that a separate high level model is not required, significantly reducing development time.

With the exception of the Design Flow 1 VHDL component, all three Design Flows have either used propriety modelling languages or propriety hardware design languages. The VHDL implementation however was written in the IEEE VHDL language standard, thus providing a substantial degree of portability between vendor tools.

In general the propriety nature of system level design languages causes concern for whole of life costs for TEP designs. A promising system design language to address this issue is SystemC.

8 RECOMMENDATIONS

Further assessment of the technical capabilities of design methods and techniques used for system designs, would be beneficial in the following areas:-

- Significantly increasing the size of design challenge, such that is no longer possible to target a single FPGA or even a single COTS subassembly card. This will enable the quantitative evaluation of design flow methodologies providing design abstraction for targeting multi-card subsystems for TEP.
- Further evaluation of the GEDAE design flow, using future versions of GEDAE able to support FPGA targets. This will allow an improved assessment of this design methodology against Design Flows 1 and 3.
- Evaluate further system design languages such as SystemC and PrecisionC, with the metrics used in this report. Of particular interest is SystemC in addressing the proprietary nature of other design flows by providing an open standard language.

9 ACKNOWLEDGEMENTS

The work described in this paper was undertaken by the BAE SYSTEMS Advanced Technology Centre under the EMRS-DTC Application Centric Processing project as part of the initial phase of a group of projects of the Transducer Embedded Processing (TEP) theme. The other projects in this group are Rapid Deployment Methods (Roke Manor Research), System Design Tools and Techniques (GEDAE Inc.), and Reconfigurable Sensor Processing (Nallatech Ltd). These are complementary activities and have been grouped together under the thematic area Transducer Embedded Processing (TEP) of the DTC-EMRS Programme to undertake research in the technical area of "System design tools and techniques".

10 REFERENCES

1. BAE SYSTEMS, Critical Requirements and the State of the Art: System Design, Tools and Techniques.
2. Nallatech Ltd, Nallatech Technical Questionnaire Results – Provisional, Project EMRS/DTC/1/139, Report No. EMRS-TQR-0.
3. IEEE 1076 Std. VHDL.
4. IEEE 1364 Std. Verilog Language, <http://www.verilog.com/IEEEVerilog.html>
5. Mentor Graphics, <http://www.mentor.com/>
6. Handel-C, www.celoxica.com
7. Open SystemC Initiative, <http://www.systemc.org/>
8. GEDAE, Dataflow model based design tool, www.gedae.com
9. Matlab Simulink, Mathematical based dataflow tool, <http://www.mathworks.com/>
10. Altera Stratix <http://www.altera.com/>
11. Xilinx Vertex, <http://www.xilinx.com/>
12. Celoxica, Timing Analysis and Optimisation of Handel-C Designs for Xilinx Chips, AN68 V1.1 Internet: www.celoxica.com/support
13. EDA Industry working group, <http://www.eda.org/>