An MDA Approach for Rapid DSP Application Synthesis over Heterogeneous Reconfigurable Platforms

J. McAllister, A. Meena, R. Woods
Institute of Electronics, Communication and Information Technology (ECIT),
Queen's University of Belfast,
Belfast, BT3 9DT, UK.

Abstract

The evolution of Field Programmable Gate Array (FPGA) devices to the current state-of-the-art System-on-Chip (SoC) devices poses considerable problems in terms of ensuring designer productivity in deploying high end military signal processing applications to these devices. This paper describes the rapid implementation and high design productivity characteristics of Owen, by highlighting the small amount of run-time required for automated generation of device programming files, and outlining a scheme for early implementation resource estimation of DSP algorithms, potentially even before a target embedded platform has been defined. This scheme achieves 90% accuracy in a matrix inversion architecture estimation, and can potentially achieve this in a significantly shorter time than current approaches.

Keywords: Embedded systems, programmable hardware, DSP implementation

Introduction

The potential for use of Field Programmable Gate Array (FPGA) as an accelerator complement to the traditional platforms for high end military Digital Signal Processing (DSP) systems comprising multiple RISC and DSP software processors has long been recognised [1-5]. However, FPGAs have more recently evolved into single chip heterogeneous multiprocessing solutions, comprising potentially of numerous software processors as a complement to the standard dedicated hardware accelerators.

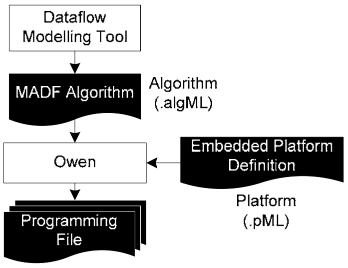
However, a significant problem exists is FPGA is to achieve it's potential as a single chip multiprocessing platform for the heart of modern DSP systems. The ongoing Owen project [6] aims to develop this technology. The paper at [6] outlines the ethics of Owen and it's place in the pantheon of embedded system design tools.

A key performance characteristic for embedded design tools like Owen is it's capability to enhance designer productivity. This paper studies Owen's capabilities in this area. Firstly, the run-time performance of the tool for a simple example implementation is outlined, focussing specifically on the added impact of using Owen over and above the other necessary automated tools for FPGA design. Given this analysis, the focus of the paper later shifts to early implementation resource estimation as an enhanced productivity feature of Owen, and analyses the accuracy of early resource estimation in Owen for a complex design example.

Owen Rapid Implementation

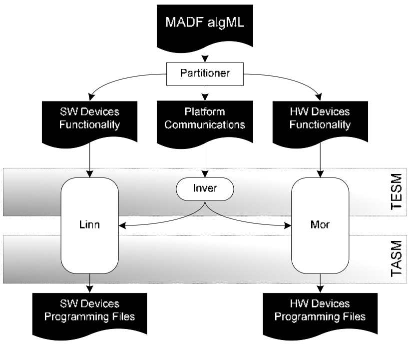
Owen is an implementation flow, and as such harnesses third party simulation technology to act as DFG development environments. The system design context in which Owen operates is shown below. As this shows, a dataflow modelling tool (e.g.

Labview, Gedae or Ptolemy) is used to define the algorithm. This tool-specific definition is then exported into a tool independent XML format (known as *algML*), which Owen accepts as its input. This input is then iteratively refined to an implementation for the chosen target platform. This conversion from tool specific formats to a tool independent markup allows the Owen implementation flow to be portable across any commercial modelling tool. However, development is focused on implementing Labview dataflow graphs.



Owen Operation Environment

Similarly, the target platform is defined in a platform description XML (known *pML*), meaning that the Owen design flow is portable across embedded platforms. The internal operation of the Owen toolset to translate the MADF algorithm specification to an implementation on an embedded platform is outlined below.



Owen Implementation Flow

The MADF algorithm representation is firstly partitioned between hardware and software. Three processes are involved in the *Owen* flow: software synthesis for software devices (*Linn*); hardware synthesis for the hardware devices (*Mor*); and Inter-Processor Communications (IPC) framework (*Inver*). They are divided between two distinct stages, Technology Specific Mapping (*TESM*) and Target Specific Mapping (*TASM*).

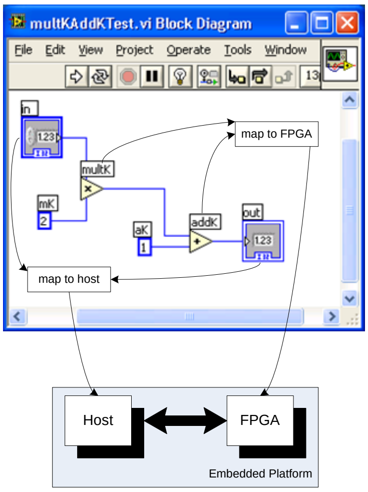
Owen currently supports two architecture templates: a point-to-point communication scheme ($P2P_0$) and a bus-based (bus_0) architecture, with each hinting at the communications protocol used for communication between nodes. In the $\mathrm{P2P}_0$ template, cores are connected via point-to-point links supporting two types of data traffic: streams and parameters. Streams are run-time bandwidth demanding streams of data resulting from computations on the input data samples. These are directly related to the FIFO channels used in dataflow modelling, and can be optimised using DFG transformations [10, 11]. The bus_0 template constructs a core network built around a centrally shared bus communications resource.

This framework enables rapid implementation and optimisation of complex DSP system applications on FPGA. To illustrate how this may be exploited and achieve different architectures whilst increasing designer productivity, consider the design of a simple multiply-add operation, as outlined next.

Owen Run-Time Performance Analysis

The exemplar application for the run-time performance analysis is a simple multiply-add operation, the Labview algorithm definition for which is shown overleaf.

The example is ported to a Xilinx University Program (XUP) Virtex2 Pro (V2P) evaluation platform available commercially from Digilent, Inc [14]. The XUPV2P platform offers the use of a Virtex-II Pro XC2VP30 FPGA via a host PC using an Ethernet communications interface. The platform structure is defined as in the *.pML* shown below (note the host is missing from this description – it is assumed available by default).



Labview multKAdd Application

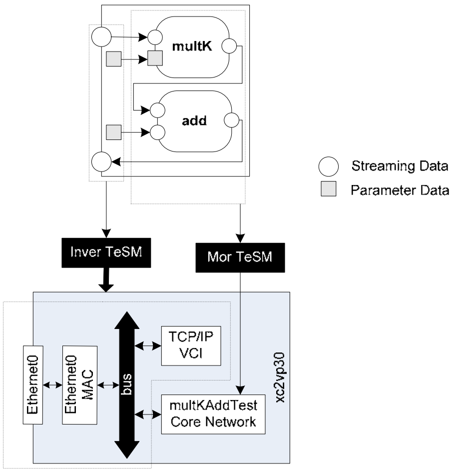
A TCP/IP software stack which operates on one of the on-chip PowerPC processors on the FPGA to offer a TCP/IP over Ethernet communications Applications Programmers Interface (API) which may be targeted by Inver allows realisation of the functionality across the heterogeneous host and FPGA components.

```
<platform>
   <device>
         <label>d0</label>
         <class>fpga</class>
         <type>xc2vp30-ff896-7</type>
   </device>
</platform>
```

xupv2p.pML

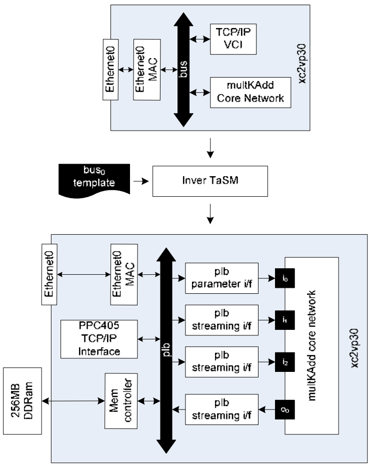
The mapping of nodes to platform resources outlined in the figure below necessitates support for streaming communications over the Ethernet link for support the data traffic flowing from *_n_10* to *_n_8* and from *_n_7* to *_n_9*. The process of refining the parts of the algorithm mapped to FPGA to a working implementation is outlined below. As this shows, the streaming off-chip communication and on-chip parameter storage are handled by Inver, and the construction of the core network to realise the desired functionality is achieved by Mor.

The architectures of different parts of the same chip can be realised using different templates. To illustrate this, this design example uses the *bus*$_0$ template to realise the communications VCI, and the *P2P*$_0$ template for the multKAddTest core network.



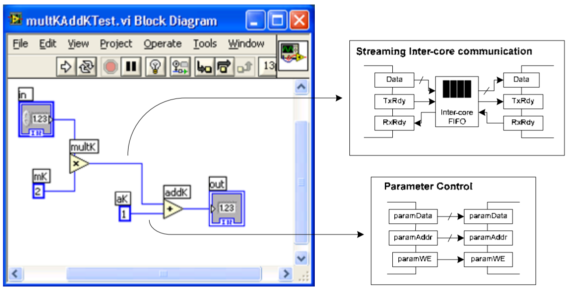
FPGA TeSM Architecture Refinement

The architecture realised using the *bus*$_0$ template in Inver, mapping the Ethernet physical to a Xilinx Core Generator Ethernet MAC core and the TCP/IP VCI to a Power PC, communicating over a *Peripheral Local Bus* (PLB) communications bus, is shown in below.



On-chip TCP/IP VCI Architecture

During the construction of the multKAdd core network in the Mor TaSM, the P2P0 template was used. This qualifies that cores are connected via point to link communication links, with different communications interface depending on the type of data traffic flowing over a link. There are two main traffic types: stream and parameter, with the differing communications protocols outlined below.



P2P$_0$ Signalling Interfaces

The run-time of the automated Owen compilation and device programming file generation routines for the FPGA are given in the table at the bottom of this page.

As this shows, the majority of the processing time spent in generating the programming files for the FPGA device is spent in third party tool (in this case Xilinx EDK). Owen itself only requires around 0.45% of the total runtime (the remainder being consumed by third party tools performing essential functions), with the input files representing the algorithm in the Labview graph available after 6 seconds of automated processing.

It is clear that to maximise the productivity of this approach, minimising the number of iterations of the TaSM is key, and any added technology at the higher levels of abstraction would be very beneficial.

One of the major architectural decisions to be made early in the design process of substantial, multi-device defence platforms regards the numbers of devices of each type required in an embedded platform. However, this presents a problem, since key resource requirements are not available until well through the TaSM phase of Mor, necessitating multiple iterations of this lengthy phase for design refinement, thus limiting the productivity of this approach. A key enhanced productivity solution therefore would be the capability to perform early (potentially before any platform exists), coarse grained estimation of the implementation resource requirements. How this is achieved in Owen is outlined in the next section.

Early Resource Estimation in Owen

The component-based design approach exploited by Owen means that relatively accurate guide resource requirements for each of the components from which a core network is built are available before implementation. By combining these to using models of the final architecture, estimation of this architecture may therefore be enabled.

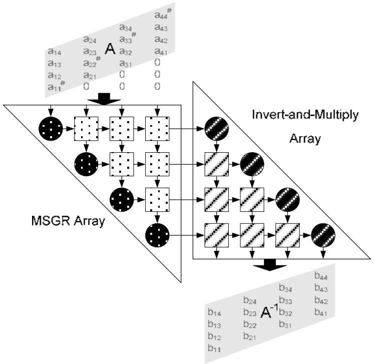
This modelling is precisely the function of the architecture template, which allows generation of a guideline architecture at the end of the TeSM phase of Mor.

Phase		Time	% Owen	% Third Party
algML Generator		1	100	0
Partitioner		1	100	0
TeSM	Inver	2	100	0
	Mor	2	100	0
TaSM	Mor	1772	0.11	99.89

Owen multKAdd Run-Time Characteristics

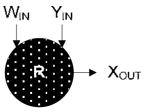
Harnessing this capability would, based on the run-time characteristics outlined in the previous section, enable early resource estimation for the implementation. The reliability of this estimation is analysed now in the context of a design example.

For complex DSP systems, matrix operations such as matrix inversion are key and exceptionally computationally complex components. New matrix inversion algorithms based on QR-decomposition [8] make this possible in real-time on a single FPGA with floating point precision. The architecture for such an operation, which may be generated directly from the algorithm specified in Labview, is outlined below.



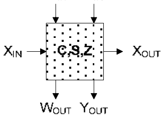
QR Matrix Inversion Array

The behaviour of each of the different cells in the array are given below.



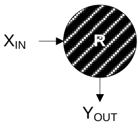
```
if ($Y_{IN}$ with flag)              //diagonal elements
    $X_{OUT}=Y_{IN}^{*}\times Y_{IN}\times W_{IN}$;
    if ($Y_{IN}$==0)                 //condition $u_k$=0
        R=1;
    else                             //condition $u_k\neq$0
        $R=Y_{IN}^{*}$;
else                                 //off-diagonal elements
    $X_{OUT}=R\times Y_{IN}\times W_{IN}$;
    R=R;
```

(a) MSGR Boundary Cell



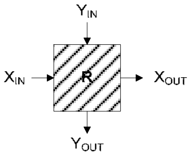
```
if ($X_{IN}$ with flag)                        //diagonal elements
    $X_{OUT}=X_{IN}+(W_{IN}\times Y_{IN}^{*}\times Y_{IN})$;
    $Y_{OUT}$=0;
    if ($X_{IN}$==0)                           //condition $u_k$=0
        C=1;  Z=0;  $W_{OUT}$=1;
        if ($Y_{IN}$==0)                       //condition $v_k$=0
            S=1;
        else                                   //condition $v_k\neq$0
            $S=Y_{IN}^{*}$;
    else                                       //condition $u_k\neq$0
        $C=Y_{IN}/X_{IN}$;  $S=Y_{IN}^{*}$;  Z=1;
        $W_{OUT}=W_{IN}\times(X_{IN}/X_{OUT})$;
else                                           //off-diagonal elements
    $X_{OUT}=(Z\times X_{IN})+(W_{IN}\times S\times Y_{IN})$;
    $Y_{OUT}=(Z\times Y_{IN})-(C\times X_{IN})$;
    C=C;  S=S;  Z=Z;  $W_{OUT}=W_{OUT}$;
```

(b) MSGR Internal Cell



```
if ($X_{IN}$ with flag)    //diagonal elements
    $R=1/X_{IN}$;
    $Y_{OUT}$=0;
else                       //off-diagonal elements
    R=R;
    $Y_{OUT}=-R\times X_{IN}$;
```

(c) Invert and Multiply Boundary Cell



```
if ($X_{IN}$ with flag)          //diagonal elements
    $Y_{OUT}$=0;
    $X_{OUT}=X_{IN}$;
    $R=Y_{IN}/X_{IN}$;
else                             //off-diagonal elements
    $Y_{OUT}=Y_{IN}-(R\times X_{IN})$;
    $X_{OUT}=X_{IN}$;
    R=R;
```

(d) Invert-and-Multiply Internal Cell

QR Array Cell Architectures

Given these behaviours expressed as flow graphs, models of the architecture resource requirements can be built, assuming that each arithmetic operation relates to a core in the architecture (note that 4 dividers required between the MSGR array and the Invert-and-Multiply array are not included in the graph above for clarity). This allows development of the models in (1) – (4) for the resource requirements of the four cell types.

$$A_{MSGR\_BC} = 6A_{f_{mul}} + 2A_{f_{add}} \tag{1}$$

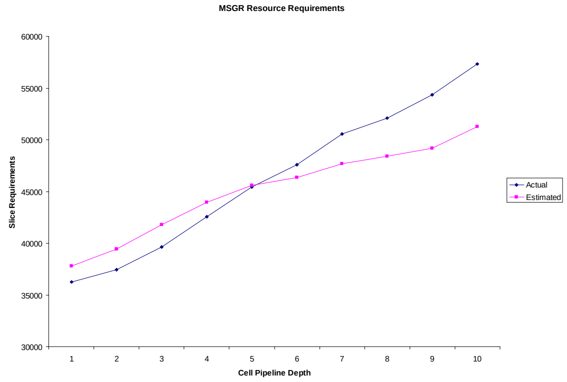
$$A_{MSGR\_IC} = 10A_{f_{mul}} + 2A_{f_{div}} + 8A_{f_{add}} \tag{2}$$

$$A_{IAM\_BC} = 2A_{f_{mul}} \tag{3}$$

$$A_{IAM\_IC} = 4A_{f_{mul}} + 4A_{f_{add}} \tag{4}$$

Using these models, in conjunction with the divider models allows estimation of the resources for the entire architecture post-TeSM. These estimations, compared with the actual resource requirements for variation cell architectures (latencies) obtained post-TaSM, are outlined in the graph and table overleaf.

As this shows, the worst case estimation error is 11% of the actual requirements. Bearing in mind the early availability of these estimates and the coarse grained nature of the desired exploration process (i.e. devices rather than slices), it is believed that this may offer a valuable enhanced productivity capability not seen before at this abstraction level.



QR Resource Estimation Results

Conclusions

The paper has described the productivity improvement capabilities offered by Owen, an MDA-compliant design tool for FPGA-centric embedded systems. Owen significantly cuts design time to translate DSP algorithms specified in graphical modelling languages such as Labview into source files for vendor specific programming tools, such as Xilinx EDK. In the simple example included in this paper, this means on 0.45% run-time overhead in converting a Labview representation of a multiply-add function into device programming files over and above the necessary implementation steps, and in a fully automated fashion. Further, it has shown how early estimation of the implementation resource requirements is achieved using Owen, eliminating productivity-constraining lengthy reiterations of low level design tasks.

References

1. Walke, RL, Smith, RWM, Lightbody, G, Advanced Signal Processing Algorithms, Architectures and Implementations X, San Diego, 2000
2. Madahar, BK, Alston, ID, Aulagnier, AD, Schurer, H, Thomas, M, Saget, B, Eurasip JASP, **2003**, 580-593, 2003.
3. Stefanov, T, Zissulescu, C, Turjan, A, Kienhuis, B, Deprettere, E, DATE, France, 2004.
4. Ying, Y, Woods, R, IEEE Trans. CAD, **25**, 806-820, 2006.
5. McAllister, J, Woods, R, Walke, R, Reilly, D, Journal of VLSI Signal Processing, **43**, 207-221, 2006.
6. McAllister, J, Meena, A, Woods, R, EMRS-DTC Conference, Edinburgh, 2007.
7. Digilent Inc Virtex-II Pro Development System. http://www.digilentinc.com.
8. Ma, L, Dickson, K, McAllister, J, McCanny, J, ICASSP, USA, 2008.

Acknowledgements

The work reported in this paper was funded by the Electro-Magnetic Remote Sensing (EMRS) Defence Technology Centre, established by the UK Ministry of Defence and run by a consortium of SELEX Galileo, Thales UK, Roke Manor Research and Filtronic.

Latency	1	2	4	6	8	10	12	14	16	18
Estimate	37780	39420	41816	43960	45608	46368	47716	48388	49196	51300
Actual	36243	37435	39617	42589	45423	47572	50585	52126	54348	57350
Error (%)	4.2	5.3	5.6	3.2	0.4	2.5	5.9	7.2	9.5	10.5

Owen QR Based Matrix Inversion Resource Estimation Accuracy