Discrete Wavelet Transform FPGA Design using MatLab/Simulink

Uwe Meyer-Baese[a], A. Vera[b], A. Meyer-Baese[a], M. Pattichis[b], R. Perry[a],
[a]FAMU-FSU, ECE Dept., 2525 Pottsdamer Street, Tallahassee, FL USA-32310;
[b]University of New Mexico, ECE Dept., Albuquerque, NM 87131

ABSTRACT

Design of current DSP applications using state-of-the art multi-million gates devices requires a broad foundation of the engineering skills ranging from knowledge of hardware-efficient DSP algorithms to CAD design tools. The requirement of short time-to-market, however, requires to replace the traditional HDL based designs by a MatLab/Simulink based design flow. This not only allows the over 1 million MatLab users to design FPGAs but also to by-pass the hardware design engineer leading to a significant reduction in development time.
Critical however with this design flow are: (1) quality-of-results, (2) sophistication of Simulink block library, (3) compile time, (4) cost and availability of development boards, and (5) cost, functionality, and ease-of-use of the FPGA vendor provided design tools.

Keywords: FPGA, two channel filter bank and Wavelet, Simulink, top down design flow

1. INTRODUCTION

Field-programmable gate arrays (FPGAs) are on the verge of revolutionizing digital signal processing. Many front-end digital signal processing (DSP) algorithms, such as FFTs, multi channel filter banks, or wavelets, to name just a few, previously built with ASICs or programmable digital signal processors, are now most often replaced by FPGAs.
The two FPGA market leaders (Altera and Xilinx) both report revenues greater than $1 billion. FPGAs have enjoyed steady growth of more than 20% in the last decade, outperforming ASICs and programmable digital signal processors (PDSPs) by 10%.

2. DESIGN CHALLENGE

System-C[1], Handel-C[2], X-Blox[3-5] or other high level tools have tried in the past to close the gap between system level design and physical FPGA implementation. However, the acceptance of these design tools has not been very favorable and they have become niche products[1-2] mainly for configurable processor design rather than as a main stream application tool, like VHDL or Verilog compiler tools on the lower implementation level. Another approach, now favored by both Xilinx[5] and Altera[6] is the design using MatLab/Simulink interface. The reason this approach is more successful is the large base of MatLab programmers of over 1 million world wide[5,6]. This design flow has several other advantages, as for instance

- Many high end FPGA applications today are in the DSP field, where MatLab/Simulink is the preferred simulation tool anyway.
- MatLab/Simulink has many state-of-the art algorithms implemented in over 25 MatLab and Simulink toolboxes, see Figs. 1 and 2.
- Simulation in Simulink can be bit precise and is an ideal framework to generate testbenches.
- The FPGA vendor provided toolboxes allow a concentration on the algorithm implementation, rather than the design tool optimizations.

The question when using such a relatively new MatLab/Simulink design flow for a designer however is the question, if this flow delivers good quality of results (QoR), or if the synthesized designs are more of a “quick and dirty” quality. Experience in the past with tools like X-Blox[3-5] showed that on average ca. 20% higher area could be expected using a

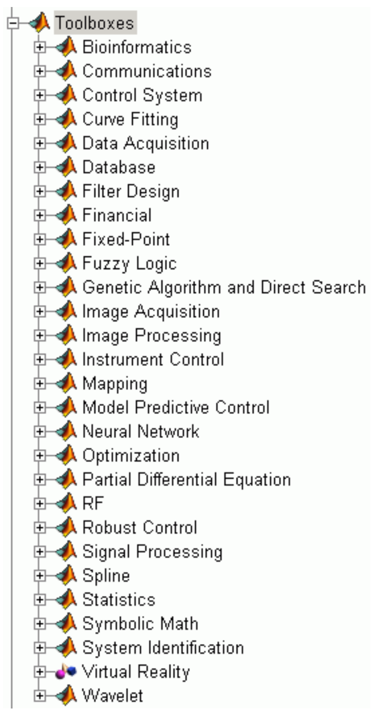


Figure 1: MatLab toolboxes.

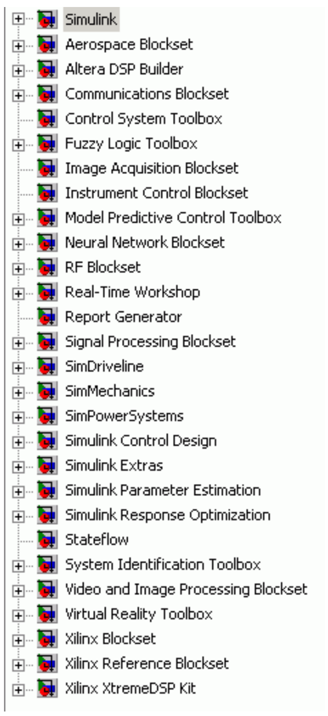


Figure 2: Simulink toolboxes.

high level tool compared with a low level HDL implementation. Addition concern of a new design flow maybe

- Available library blocks and organization
- Development board support
- Design implementation “easy of use”
- MatLab/Simulink design support
- Tools and board costs
- Quality of results (QoR)

Although most FPGA vendors offer now free web based software for download, we found that in order to use the MatLab/Simulink design interface it is required to have a “full” tool subscription, the web-based version does not support DSP builder (provided by Altera) or System Generator (provided by Xilinx). As examples, we used several different design examples that are documented in a DSP with FPGAs[7] book.

3. DEVELOPMENT TOOL STUDY

In the following we will compare the Altera and Xilinx based design flows and describe the strengths and weaknesses of the two different approaches and give a conclusive evaluation (i.e., even, or advantage Altera/Xilinx). Our evaluations are based on recent versions of the tools: Altera’s DSP Builder Version 5.0 and Xilinx’s System Generator Version 7.1.

3.1 Library Blocks : Even

The Altera and Xilinx libraries with 171 and 84 blocks are arranged in 11 and 16 subgroups respectively, see Figs. 3 and 4. (Note that the larger number of the Altera blocks comes from the ca. 20 blocks for each development board.) Both

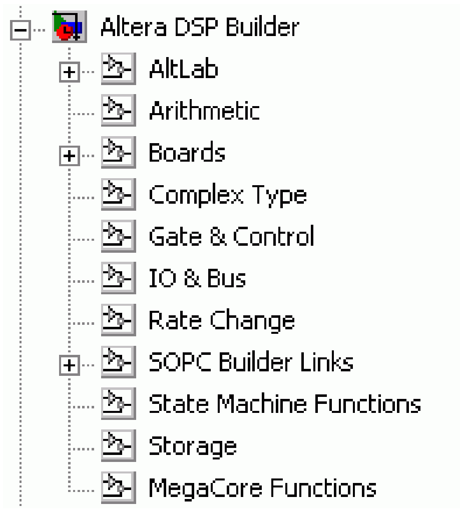


Figure 3: Altera Simulink Library.

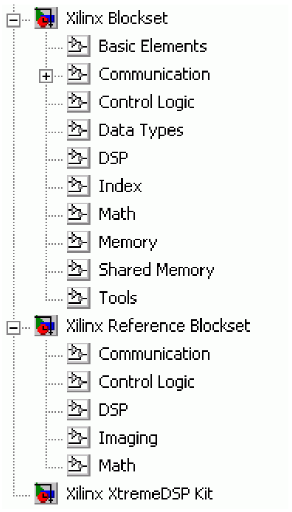


Figure 4: Xilinx Simulink library.

libraries have a large number of DSP basic building blocks, that allow the design of almost every DSP system without the need of any custom block that utilized MatLab code or HDL "black box" imported files. In fact based on the examples from the DSP with FPGAs textbook[9] all ca. 50 designs have been built without any custom MatLab script or black-box HDL code using only Xilinx and Altera predefined Simulink blocks. Both collections of library blocks have strengths and weaknesses. Altera has some more useful basic blocks, like a case block, or decoder block. Xilinx provides some more sophisticated Intellectual Property (IP) blocks like FFT, CORDIC, FIR that are included and do not require additional licenses as with the Altera software. Since both libraries have distinct strengths, the overall evaluation is even in this category.

3.2 Library Organization : Advantage Xilinx

The Xilinx block libraries are well organized in subcategories, Altera's DSP Builder blocks are less easy to find and sometimes have strange associations. The Altera `downsampling` and `upsampling` blocks for instance are *not* in the multirate subgroup, you find them in the "storage" group. Adder blocks are not listed first in the "arithmetic" subgroup, you find them under `Parallel Adder Subtractor` or `Pipelined Adder`. Just changing the sequence to "`Adder Parallel`" and "`Adder Pipelined`" could potentially save many designers time to find these blocks. Also missing with the Altera library is an "Index" subgroup, that contains all blocks listed in alphabetic order, which is available in the Xilinx System Generator environment. This can save valuable time if one does not like to scan all subgroups for the block that one is looking for, especially when the subgroup title does not reflect the contents;

XILINX®
ADVANTAGE

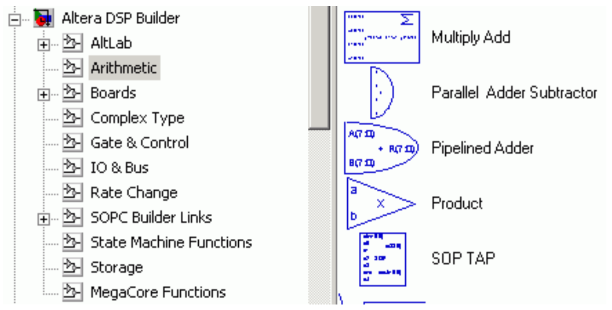


Figure 5: Altera adder blocks.

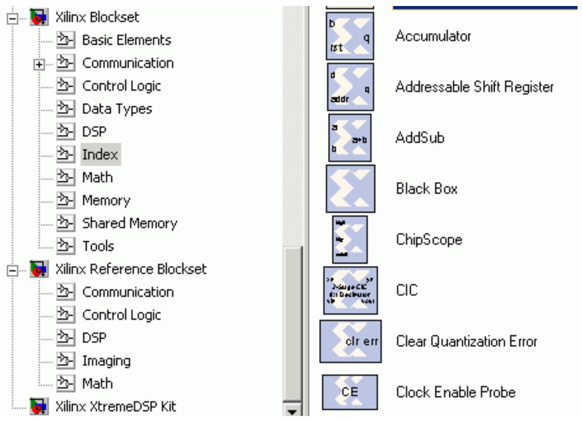


Figure 6: Xilinx Index subgroup.

as for the Altera `downsampling` and `upsampling` blocks. Overall, our evaluation is therefore in favor of the Xilinx library organization.

3.3 Simulink Design Support : Advantage Xilinx

When it comes to developing a design with Simulink blocks, the Xilinx system generator library has a couple of features that gets one up to speed much faster than with the Altera tools. Our experience with ca. 50 designs using Xilinx and Altera predefined Simulink blocks showed, that it took on average twice as long to get the Simulink simulation to work with the Altera software compared to the Xilinx environment. Let us have a brief look at four different examples from data type, control signals, pipelining your design to Simulink symbol manipulation:

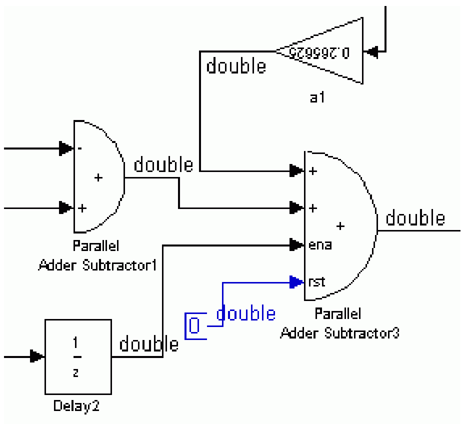
XILINX®
ADVANTAGE

a) **Data types**: If one turns on the option **Port Data Types** under the **Format** menu and **Port/Signal Display** submenu in Simulink, one is able to see the actual data types used in Simulink between the blocks. The Altera blocks are all from the double data type, the exact I/O bit width used by a block can only be seen when you look inside a specific block, see Fig. 7. On the other hand, Xilinx blocks and the associated connecting nets show the exact bit width and data type specified, see Fig. 8. This can save a lot of time in order to get a design to work properly.

b) **Control signals**: Most blocks offer additional data or control signals, like enable or reset signals, or carry out. In the Xilinx blocks each additional data/control signal can be separately turned off or on, and only the control signals that are chosen are displayed in the block diagram. On the Altera blocks one can only select to use all control signals or no control signals at all. One then also needs to connect the additional control signals to the appropriate sources (GND/VCC) or sinks (terminators) in case one does not need the signals. This not only adds unnecessary complexity to the design, but it is also required to use the right polarity for the additional control signal that wasn't really needed. These I/O blocks cannot be simply left unconnected, since there is no default assignment like in the VHDL LPM component that can be used. Simulink will report an error for all unconnected ports.

c) **Pipeline the design**: Most of the blocks allow one to introduce additional register delays to a block, which is usually called pipelining, in order to increase the throughput of the designs. In the Xilinx blocks these additional delays are displayed using a z^{-d} symbol inside the block, see Fig. 8, while Altera blocks (beside the delay block itself) do not show the additional added delays. One needs to look inside the block to identify the number of delays in each Altera block.

d) **Symbol Block representation**: If one flips the direction of a Xilinx block one is still able to read the associated numbers. In the Altera blocks the numbers are now upside down and much harder to read, see Fig. 7.



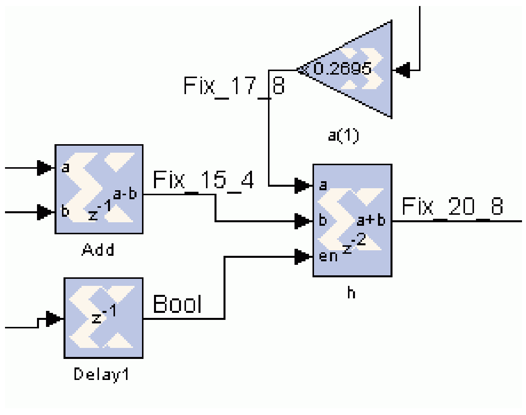


Figure 7: Altera ports and nets.

Figure 8: Xilinx ports and nets.

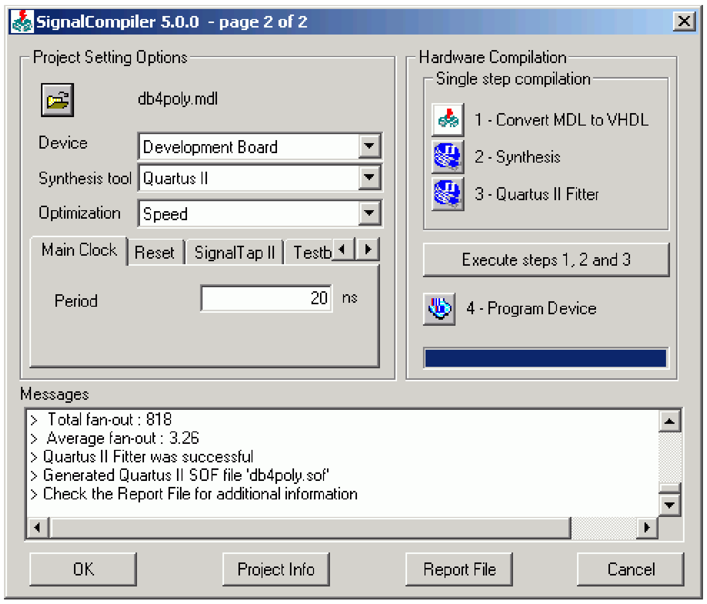
All four features in handling the design support are more user friendly in the Xilinx designs and our evaluation is therefore in favor of the Xilinx design support. It took us on average twice as long to get the Simulink simulation to work properly with the Altera blocks compared to the Xilinx environment.

3.4 Design Flow Implementation : Advantage Altera

In the Altera design flow implementation one can call DSP Builder within Simulink and start the full compilation processor: MDL→VHDL, Synthesis, Fitter and board programming within a single GUI. Report files are linked to the Simulink block that allows one to study details of resource usage as well as the timing information. All data are available in Simulink and can be accessed with a few button clicks, and there is no need to start low level compiler like Quartus II by hand. This is different with the Xilinx flow. Here one can generate the VDHL code using System Generator, but then one needs to be very familiar with the ISE software. To incorporate the pin assignment of the I/O blocks like switches or A/D converter, for instance, one can manually link the *.ucf file, or modify the file <design>.xcf created by Sysgen. A better choice is probably to build all "custom block" for the components on the Digilent board in Simulink that already has a specific I/O assignment. After compilation of the HDL code one then needs to use iMPACT to program the board. The Xilinx design flow is far away from being a push button approach and that maybe for a software only developer too much to ask for. This is a critical issue, since the main target group of this flow are the MatLab/Simulink software developers with little or no knowledge of FPGA/ASCI tools.

ADVANTAGE

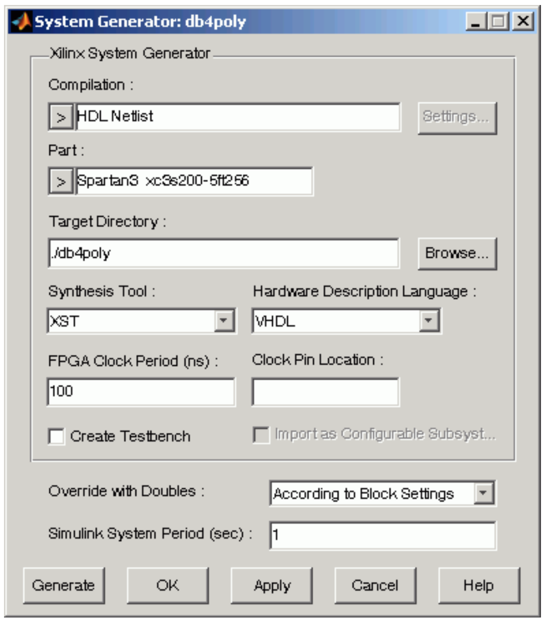
Overall, our evaluation is therefore in favor of the Altera Design flow, which can save valuable time in the design process.



db4poly.mdl
Device: Development Board
Synthesis tool: Quartus II
Optimization: Speed
Period: 20 ns

```
> Total fan-out : 818
> Average fan-out : 3.26
> Quartus II Fitter was successful
> Generated Quartus II SOF file 'db4poly.sof'
> Check the Report File for additional information
```

Figure 9: Altera Simulink to FPGA path.



Compilation: HDL Netlist
Part: Spartan3 xc3s200-5ft256
Target Directory: ./db4poly
Synthesis Tool: XST
Hardware Description Language: VHDL
FPGA Clock Period (ns): 100
Clock Pin Location:
Override with Doubles: According to Block Settings
Simulink System Period (sec): 1

Figure 10: Xilinx Simulink to FPGA path.

3.5 Development Board Support : Advantage Altera

Xilinx does not provide its own collection of development boards, only for one XtremeDSP kit, a few blocks are available for A/D and D/A that can be used, see Fig. 12. For any other boards, e.g. the Digilent XC3S200 boards a popular board available through the University program, no such blocks are available and have to be generated by the designer.

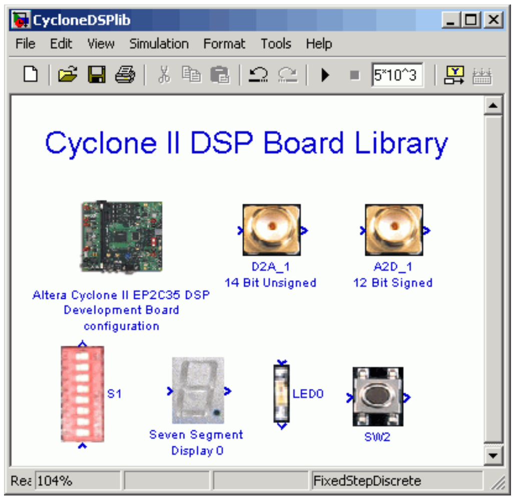

CycloneDSPlib
File Edit View Simulation Format Tools Help
Cyclone II DSP Board Library
Altera Cyclone II EP2C35 DSP Development Board configuration
D2A_1
14 Bit Unsigned
A2D_1
12 Bit Signed
S1
Seven Segment Display 0
LED0
SW2
104%
FixedStepDiscrete

Figure 11: Altera board support: 7 out of 19 blocks.

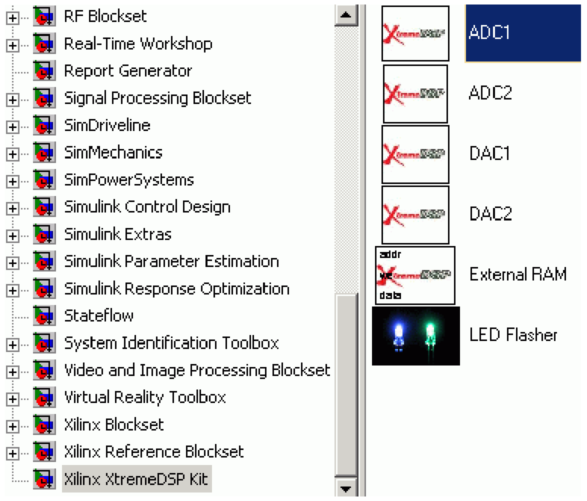


Figure 12: Xilinx board support (all).

One has to find out the pin assignments and create the custom blocks, which maybe too much to ask from a pure software MatLab developer.

Altera supports 4 DSP development boards with a full set of components in Simulink, including A/D, D/A, DIP switches, LEDs, and push buttons, see Fig. 11. These components can be directly used in Simulink and are wired in the correct way on the boards. One "configuration" blocks is used to tell DSP builder the used device and pin assignment for clock and reset. In SignalCompiler one simply selects development board as device and all data from the configuration block are used. A small disadvantage is that with Altera development boards currently the on-chip PLL can not be used to support multiple clock domain and for multirate design, a master clock and appropriate clock enable have to be applied.

ALTERA
ADVANTAGE

Overall there is much more support from the Altera side for the boards given, and our evaluation is therefore in favor of the Altera board support.

3.6 Money : Even

The financial requirement to build a DSP with FPGAs hands on experience for designer or students usually has two components: The software and hardware. Although one may be able to request some or all of the hardware/software

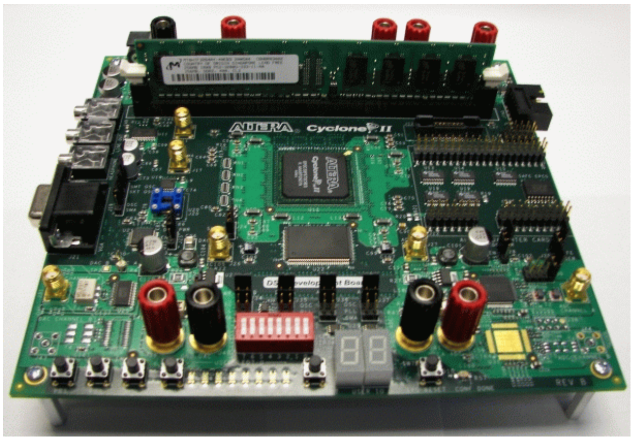


Figure13: Altera low cost board.

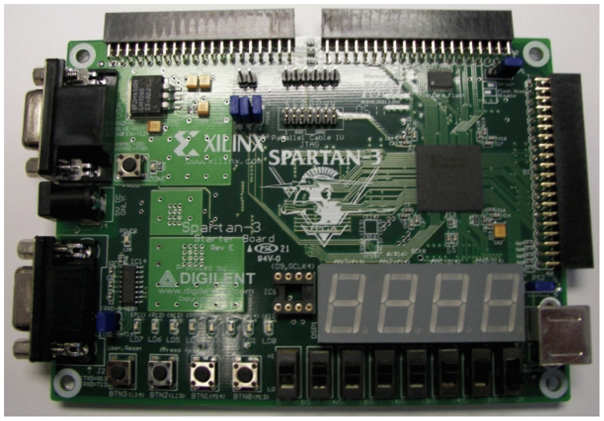


Figure 14: Xilinx low cost board.

parts to be donated, or used in an evaluation mode, this is not always the case. Let us have a look at University pricing to run a laboratory up to 10 years with up to 10 stations. The required Altera software tools (Quartus and DSP Builder) come free of charge without even any maintenance fees. Xilinx tools require an annual fee of $250 for ISE and $359 currently for System Generator. In addition, MatLab/Simulink licenses are required. Pricing depends on the terms of license the University has with MathWorks, which is for instance at FSU (see, http://sl.us.fsu.edu/matlab.html) $36.73 annually for MatLab/Simulink and 7 tool boxes (Control System Toolbox, Image Processing Toolbox, Wavelets Toolbox, Signal Processing Toolbox, Statistics Toolbox, Optimization Toolbox). In the worst case, each student/group has to purchase a student version priced today at $99 each.
A nice set of different development boards is available from Altera directly. They have Stratix S25, Stratix II S60,and S80 and Cyclone boards, in the $700-$5995 price range ($700 University price Cyclone II board), which differs not only in the FPGA size, but also in the extra features, like number of A/D channels and memory blocks. For Universities a good choice will be the lowest cost Cyclone board, which is still more expensive than a UP2 or UP3 board used in many digital logic labs, but has a fast A/D and D/A and a two channel CODEC, and large memory bank outside the FPGA, see Fig. 13.
Xilinx on the other side has a very limited direct board support, all boards for instance available in the University program are from third parties. However some of these boards are priced so low that it seems that these boards are “not-for-profit” designs. A good board for DSP purpose (with onchip multiplier) is for instance offered by Digilent Inc. for only $99. The board has a XC3S200 FPGA, flash , 4 LEDs, 8 switches, and 4 push buttons. For DSP experiments, A/D and D/A mounted on very small daughter boards, are available for $19.95 each, so a nice DSP board can be build for only $138.90. Unfortunately, there is no Simulink block support for the board, which should be built by the lab TA/instructor.
A break down assuming 1-5 year lifetime of the lab and the lowest cost boards available today is shown in Fig 15. The total sum includes FPGA software, hardware, TA and MatLab/Simulink costs. It can be seen that for fewer stations the break even point are reached early, for 10 stations the break even point is after 9 years, which is probably longer than the lifetime of the lab. We also added the cost for MatLab and a TA/10 h/week. The Xilinx lab offers a cost advantage for many stations or short lifetime, in the other cases the Altera Lab is more cost efficient.

Since both solutions are very similar in costs, the overall cost evaluation is even in this category.

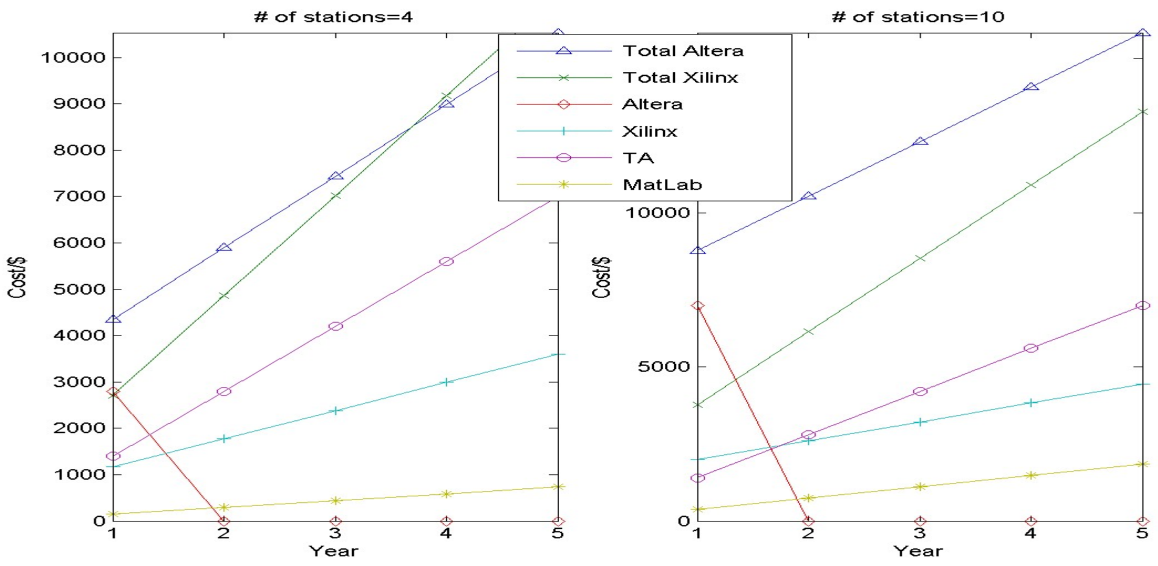


Figure 15: Cost break even points for Altera and Xilinx using 4 and 10 stations, respectively.

3.7 Quality of Results : Even

When a new design flow is considered, the Quality-of-results (QOR) is of main concern. We may argue that at a University lab one does not care much about the quality of the design results, since most of the designs are small and for demonstration purposes of the concept only, and probably will not be used in the same setting in an industry application. On the other hand, we need to be concerned about QOR, because if QOR is not acceptable, the whole design flow can be questioned. As a consequence, we should not teach such a flow in a University lab, or use this design approach in industry. In order to justify that the Simulink FPGA design flow gives good QOR we have therefore evaluated several different examples from the DSP with FPGAs[9] book, that allows us to compare this high level design with hand optimized HDL code. The overall performance data in terms of size and speed are shown in Fig. 16 for 6 different examples:

1. Lattice Daubechies length 4 two channel filter bank (db4latti)
2. Polyphase filter bank for Daubechies length 4 filter (db4poly)
3. 2 TAP delayed by 6 LMS filter (fir6dlms)
4. Length 4 programmable FIR filter in transposed form with loadable coefficients (fir_gen)
5. Two TAP LMS filter (fir_lms)
6. Length 4 direct form FIR filter for 4 fixed fractional coefficient (fir_srg)

Design	**Altera Cyclone II Board**						**Xilinx Digilent XC2S200**					
	VHDL			Simulink			VHDL			Simulink		
	LCs	Mul.	MHz	LCs	Mul.	MHz	LCs	Mul.	MHz	LCs	Mul.	MHz
db4latti	404	0	30.6	206	6	81.4	584	0	28.56	345	0	39.8
db4poly	168	0	109.0	102	2	189.8	165	0	86.68	111	0	118.6
fir6dlms	115	4	170.9	129	4	177.6	42	4	91.7	64	4	110.6
fir_gen	120	4	250.0	122	4	329.7	57	4	86.4	57	4	86.9
fir_lms	50	4	65.5	57	4	65.1	42	4	42.3	64	4	48.5
fir_srg	110	0	92.0	84	0	155.6	118	0	49.95	50	0	85.7
Average	161.2	2.0	119.6	116.7	3.3	166.5	168.0	2.0	64.3	94.7	2.0	81.7
Gain %:				38.1	-40.0	39.2				77.3	0.0	27.1

Figure 16: QOF for VHDL ↔ Simulink comparison.

It can be seen from Fig. 16 that the size and speed data for HDL and Simulink design are very similar, sometimes the Simulink design is even better than the hand optimized HDL, with the additional benefit to have a fast implementation and verification/test bench at no extra design time/cost. The only negative gain was noticed for the Altera tools use of embedded multipliers. The tools have the tendency to use more multipliers than the HDL based designs if we optimize the designs for speed. Overall both FPGA vendor tools perform pretty well and our overall evaluation is therefore even in the QOR category.

Design	**Altera DSP Builder 5.0**					**Xilinx System Generator v7.1**		
	MDL=>			Timing		MDL=>	Fitter +	
	VHDL	Synthesis	Fitter	Analysis	**Total**	VHDL	Synthesis	**Total**
db4latti	0:02	0:11	0:26	0:02	**0:41**	0:15	1:24	**0:49**
db4poly	0:02	0:11	0:23	0:02	**0:38**	0:55	1:14	**1:04**
fir6dlms	0:02	0:10	0:25	0:02	**0:39**	0:52	1:11	**1:01**
fir_gen	0:02	0:08	0:26	0:02	**0:38**	1:07	1:33	**1:20**
fir_lms	0:02	0:09	0:24	0:02	**0:37**	0:28	1:04	**0:46**
fir_srg	0:02	0:07	0:21	0:02	**0:32**	0:47	0:59	**0:53**
Average	0:02	0:09	0:24	0:02	**0:37**	0:44	1:14	**0:59**

Figure 17: Compile time in seconds using Simulink → VHDL interface.

Last, but not least, we also evaluated the compile time, since for such a ca. 20K equivalent gates design an ASIC synthesis using Synopsys `Design_Analyzer` may require up to one hour. The compile time for a lab is a critical parameter since the student should complete the lab experiment within a reasonable time. Fig. 17 show the measured compile time for a Dell 2.66 GHz PC with 1.5 GB main memory running MS Windows XP. The average compile time is very reasonable requiring less than a minute, and the students will be able to complete the experiments within the lab time.

4. ACKNOLEGEMENT

The authors acknowledge NSF support under CCLI award DUE-0442600. The authors would also like to thanks Altera and Xilinx for the provided hardware and software under the University programs. The help of the following research assistants, consultants and program manager is acknowledged: B. Esposito, M. Phipps, R. Maroccia, (Altera) J. Weintraub, A. Acevedo, C. Sepulveda, C. Dick (Xilinx) P. Athanas, (Virgina Tech), A. Hoover (FSU), E. Kim (FSU), H. Hamel (Raytheon), W. Moreno (USF), S. Connors (Boeing-SVS), J. Hallman (Harris Corp.), E. Manolakos (Northeastern University), E. Zurek (USF) .

5. CONCLUSION

We have demonstrated that the Simulink design flow for FPGAs has become a viable design path. Both vendors provide a sophisticated toolbox of library elements that are almost complete and the quality of results (QOR) for most of the Simulink design matches or exceed hand optimized VHDL coding. The compile time (for both tools sets on average below a minute for a 20K gate design) seems very reasonable. The design flow allows a software developer to quickly explore FPGA design options in terms of size and speed and to check if the resulting design fulfills the design constrains. Overall the comparison showed the strength of the tool: Xilinx strength was in “Simulink Design Support” and “Library Organization.” Altera tools were best when it came to “Development Board Support” and “Design Flow Implementation.” Even evaluations were reached for “Library Blocks,” “Money” and QOR. We have also identified some weakness: Xilinx tools for instance are much easier to handle on the Simulink design level, while Altera tools offer a design flow controlled by Simulink, from MDL→VHDL conversion to the device programming, not available with Xilinx tools. The costs for a development lab are reasonable, given the fact that software (MatLab, FPGA tool) as well as hardware boards are available at University prices. In conclusion, the Simulink design flow for FPGAs is an interesting alternative both for a University lab as well as for the professional developers in industry.

REFERENCES

[1] Handel-C homepage: www.celoxica.com/

[2] System-C homepage: http://www.systemc.org/

[3] U. Meyer-Baese, A. García, and F. Taylor. "Implementation of a Communications Channelizer using FPGAs and RNS Arithmetic." Journal of VLSI Signal Processing, Vol. 28, pages 115-128, May 2001

[4] S. Dworak, "Design and Realization of a new Class of Frequency Sampling Filters for Speech Processing using FPGAs" Master's Thesis, Institute for Computer Engineering, Darmstadt University of Technology, 1996.

[5] Xilinx XBLOX and System Generator http://www.xilinx.com/

[6] Altera DSP builder: http://www.altera.com/support/ip/dsp/ips-dsp-builder.html

[7] The MathWorks Inc., "Learning MATLAB 7", Release 14, ISBN 0-9755787-0-9, *July* 2004.

[8] The MathWorks Inc., "Learning SIMULINK 6", Release 14, ISBN 0-9755787-3-1, *October* 2004.

[9] U. Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Arrays", *Springer*, *Heidelberg* 2004.