UDRC Summer School

Industry Day
27th June 2014

LSSC WP5: PIs: Stephan Weiss, Ian Proudler (LU), RA: Keith Thompson, PhD: Jamie Corr

Overview

1) LSSC WP5 - Low Complexity Algorithms and Efficient Implementation

RA: Keith Thompson

‘Outline of Mathworks Tools used in LSSC WP5’

2) Mathworks UK (Cambridge)

Marc Willerton, Applications Engineer

Part of UDRC 1 – Imperial College

‘Efficient System Level Simulations and Model Based Design using MATLAB and Simulink’

LSSC WP5 links with Mathworks

- Original LSSC WP5 industrial partner Steepest Ascent acquired by Mathworks in Autumn 2013.
- Formed ‘Mathworks Glasgow’ with strong links with University of Strathclyde (Prof. Robert Stewart, CIDCOM research group)
- Wireless Communications Application Development - LTE System Toolbox^{TM}
- Hardware Development on FPGAs, Zync SoC, industry links with Xilinx
- PhD students working between Strathclyde and Mathworks
- Desire to build further links with Mathworks Cambridge

LSSC WP5 Overview

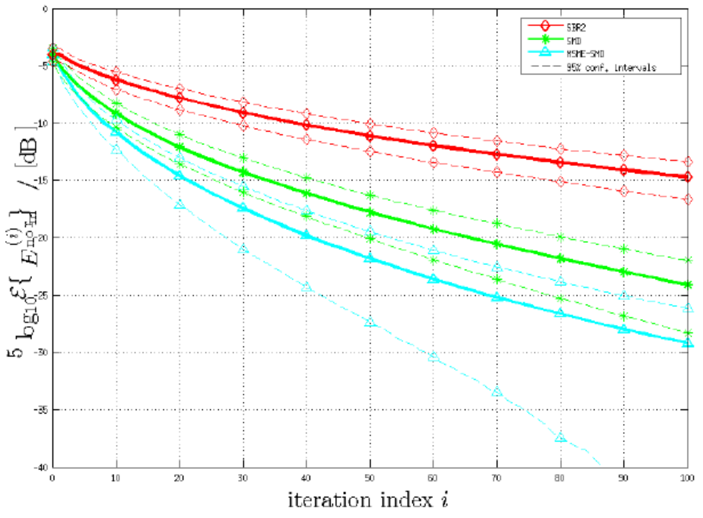
- WP5.1) - Low Complexity Algorithms and Distributed Processing
- WP5.2) - Hardware Realisations

- WP5 is to support development of hardware and efficient methods across all WP1 – WP4.
- Relates to [dstl] Challenge 29 - "Reducing Size, Weight and Power Requirements Through Efficient Processing"

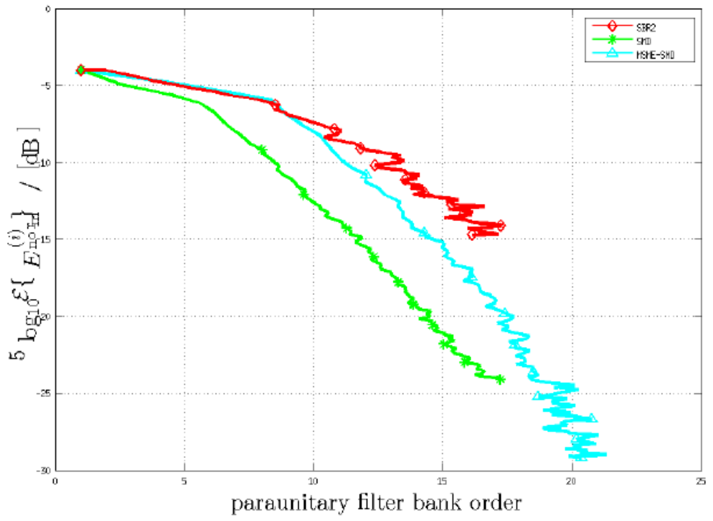
- Objective - Reduce “Implementation Gap” of algorithms developed in LSSC

WP5.1) Low Complexity Algorithms

- Sub-space Techniques - Polynomial Eigen-value Decomposition (PEVD) used in Source Separation applications
- Recent work (SMD, MSME-SMD) improves performance of iterative methods to diagonalise space-time covariance matrix.
- New methods transfer more energy per step, further benefits of reducing filter-bank requirements, and improved spectral majorisation. But more computationally demanding!



(1) Convergence Plot of Diagonalisation



(2) Diagonalisation vs. Filter-bank Order

Improving Alg. Efficiency in Matlab

- Building a ‘PEVD Toolbox’ of Optimised Matlab to be made available to consortium - include different algorithms, examples
- Optimise Matlab code with ‘Profiler’ - provides execution speed breakdown of functions called, useful to identify ‘bottlenecks’ (e.g. vectorise code, preallocate matrices, Column-wise processing etc.)
- Re-factor code into Entry-Point function, Test scripts
- Accelerate Matlab code with MEX compiler
- Accelerate using Parallel Processing Toolbox (identify parallelisation) – can use multicore CPU, GPU, local PC network
- Generate Documentation from Matlab ‘M2HTML’: http://www.artefact.tk/software/matlab/m2html/

WP5.2) Hardware Realisations

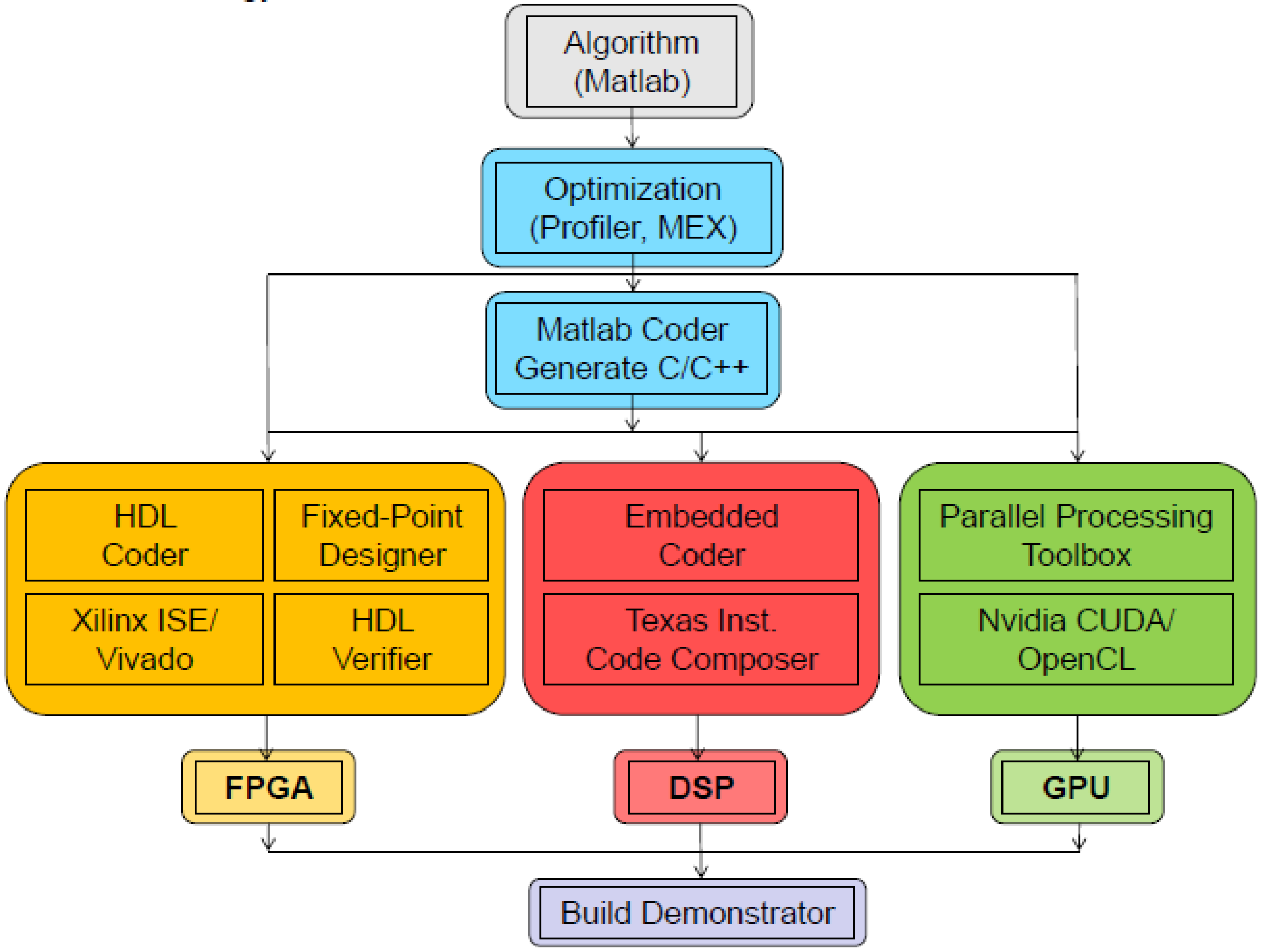
- View MATLAB algorithm and Toolbox Development as the starting point (Profile, Refactor Code, Alg. Acceleration)

Further Stages in MATLAB:

- Simulink – Useful for 'System Level' Design to modify MATLAB code into 'User-Defined Function Blocks', e.g. convert algorithm into serialized designs for analysis of large datasets
- Matlab Coder – Generate C code for DSP, SoC Hardware
- HDL Coder – Generate HDL code (VHDL/Verilog) for FPGA Hardware
- Fixed-Point Designer – Optimise word-lengths for non-floating point processors, FPGAs
- CUDA (C-based) for processing on GPU Hardware

WP5.2) Hardware Realisations

General WP5 Strategy:



WP5.2) Code Generation

- Code Generation Advantage - obviously don't have to write code from scratch!
- More hardware-friendly algorithm remains in Matlab development environment, thus easier to gauge performance (use visualisation tools) and reference against original algorithm (keeps designer in the loop). Note: 'Matlab Engine' can be called from Visual Studio C++ to use Matlab Visualisation tools.

But!:

- Can't just press 'Build'! – require further optimisation and manual configuration of C/HDL, targeted at specific hardware
- Some Matlab Functions not supported – find alternative libraries, write C/HDL code.

WP5.2) Further Optimisations

1) Execution Speed Optimisation

- Inline functions, loop fusion, constant folding, loop un-rolling, remove unused execution paths, disable support for non-finites, overflows etc.

2) Memory Usage Optimisation

- Minimise Dynamic Memory Allocation, set thresholds or upper bounds, manage data types (Fixed Point Conversion), remove unused execution paths

3) Going Further:

- Replace Run-time Computation – Look-up Tables
- Use methods for Fast Evaluation of Elementary Functions – 'Shift-and Add' algorithms (CORDIC, BKM), Polynomial Approximations etc.
- Use 'OpenMP' API for Parallelization, Message Passing Interface (MPI) for distributed processing

WP5 Summary

- Develop optimise Matlab Toolboxes
- Build capability to implement algorithms onto variety of hardware targets – FPGA, DSPs, GPUs, SoC
- Hardware Laboratory

Current Implementation work:

- Implementation of PEVD Methods on FPGA
- Implementation of SAR Change Detection algorithm (from WP4) on Texas Instruments DSP board.

- Grow further links with Mathworks and Texas Instruments