University Defence Research Collaboration (UDRC) Signal Processing in a Networked Battlespace

L_WP5: Low Complexity Algorithms and Efficient Implementation

WP Leaders: Stephan Weiss (University of Strathclyde), Ian Proudler (Loughborough University)
Researcher: K.Thompson (University of Strathclyde) PhD Student: J. Corr (University of Strathclyde)

Aim: To develop novel paradigms and implementation strategies for a range of complex signal processing algorithms operating in a networked environment. Support all WPs in development of efficient methods and hardware implementations. Scientifically Possible → Technically Feasible

L_WP5.2 Efficient Implementations and Hardware Realisations

Numerically efficient algorithms to be mapped onto suitable processing platforms demonstrating sample implementations and real-time performance in suitable test scenarios.

Objective: In collaboration with industrial partners (Mathworks, Texas Instruments, Prismtech) we are establishing a Hardware Suite in a UDRC Laboratory in Strathclyde University. Hardware Suite is to build capability to implement algorithms on a variety of suitable hardware platforms:

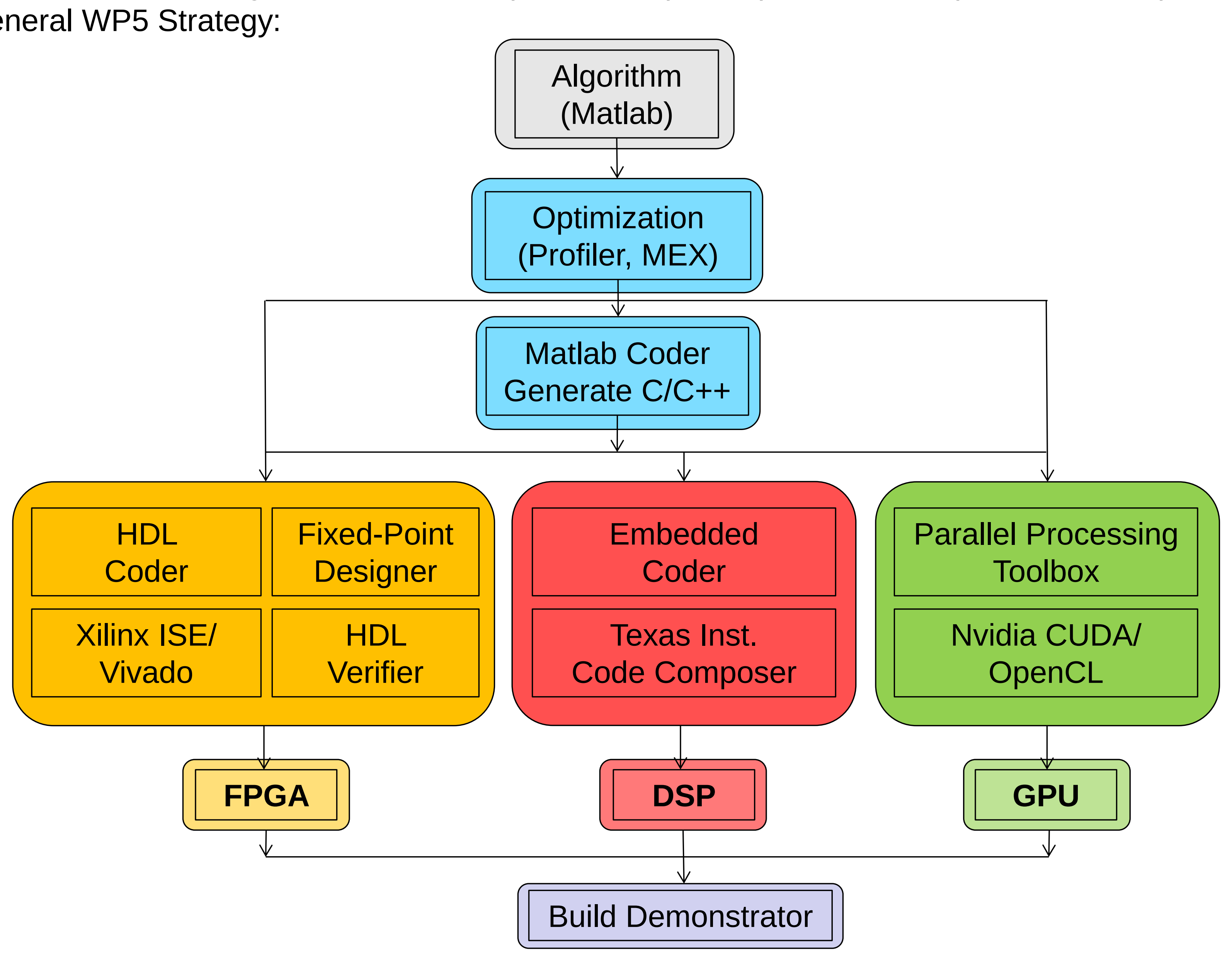
- FPGAs
- Graphic Processing Units (GPUs)
- DSP Processors

- Examine processing capability of different platforms
- Investigate Size Weight and Power (SWAP) implications
- Examine potential for Distributed Processing

Select Best Platform for Implementation

Matlab Algorithm Optimisation and Acceleration

- Matlab Best Coding Practices
  e.g. Vectorization, Pre-allocation, Profiling
- Parallel Processing Toolbox (Multicore, GPUs, Clusters) to accelerate simulation and assess parallelisation potential of algorithms
- Integration with other languages (Matlab Coder) to produce C/C++, MEX functions
- C/C++ Code allows greater execution speed and opens up hardware implementation options
- General WP5 Strategy:



Implementing Parallel Processing

- Parallel processing involves the simultaneous use of multiple CPUs, processor cores, or co-processors (e.g. GPUs) to execute a program or multiple computational threads.
- Parallel Processing Toolbox allows parallelism of Matlab algorithms to be easily explored.
- Multi-Threading has been enabled in many toolbox functions (e.g. Signal Processing Toolbox)
- For parallelisation, code must be suitable – independent tasks or iterations, data dependencies must be identified, communication between 'workers/cores' must be enabled and understood.
- Advanced and easy-to-use programming constructs for parallel execution, e.g. for loops ('parfor'), and execution of code components in parallel on workers of defined parallel pool ('spmd', 'batch').

Implementation of Parallel Processing
Parallel Processing Toolbox is being used to accelerate execution of algorithms in PEVD Toolbox initially using multiple core CPUs then GPU hardware.

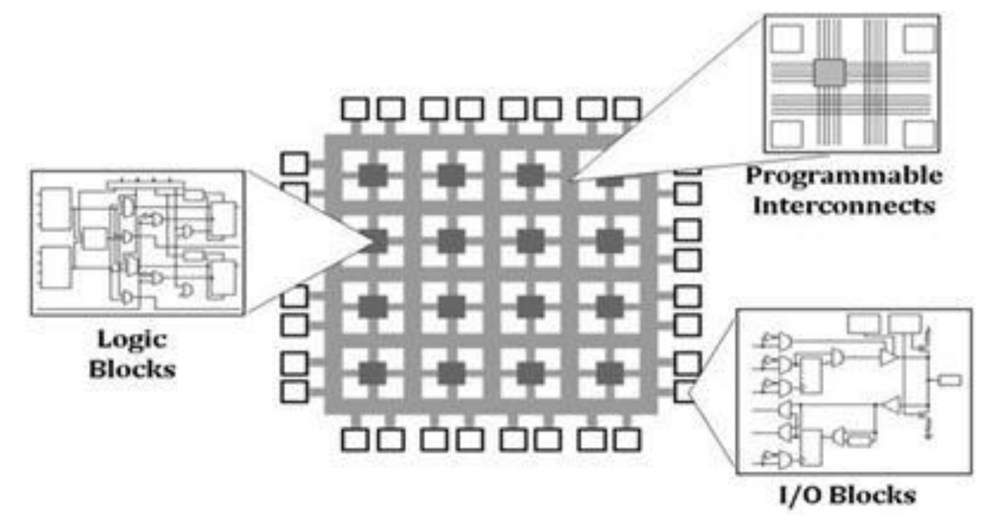
Europractice Service www.europractice-ic.com

- EUROPRACTICE is an EC initiative which aims to stimulate the wider exploitation of state of the art microelectronics technologies by European industry.
- The EUROPRACTICE Software Service provides access to a range of cost effective leading edge IC, FPGA, MCM and Microsystem design tools for non commercial teaching and research. Managed by the Microelectronics Support Centre, Rutherford Appleton Laboratory, UK.
- The EUROPRACTICE *IC* Service brings ASIC design and manufacturing capability within the technical and financial reach of small companies/researchers that wish to use ASICs. Managed by IMEC and Fraunhofer, offers low-cost ASIC prototyping, small volume production ramp-up to high volume production through Multi Project Wafer - MPW - and dedicated wafer runs.
- **Objective:** Further investigate opportunities for designing custom ASICs and evaluate costs, lead-times, design requirements.

FPGA Development

- Latest implementation tools allow HDL code (VHDL or Verilog) to be generated automatically from C/C++ and Matlab. Examining latest development tools from Mathworks (HDL Coder), Xilinx ISE (System Generator and Vivado), Mentor Graphics (ModelSim) and other vendors (Cadence, Synopsis etc.)

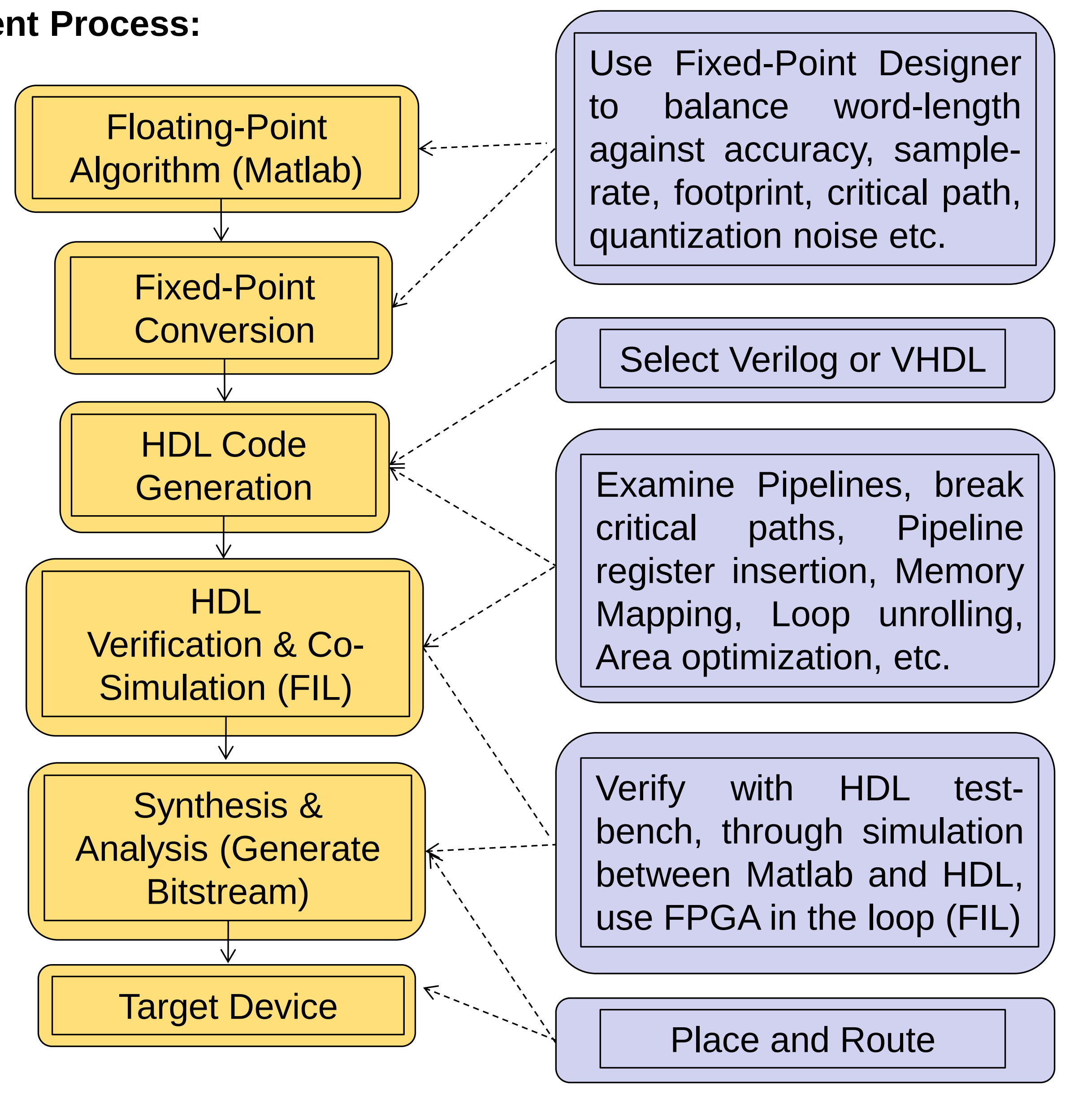
FPGA Application in Development
Implementation of PEVD algorithms (SBR2 and SMD) on FPGA hardware.



from ni.com

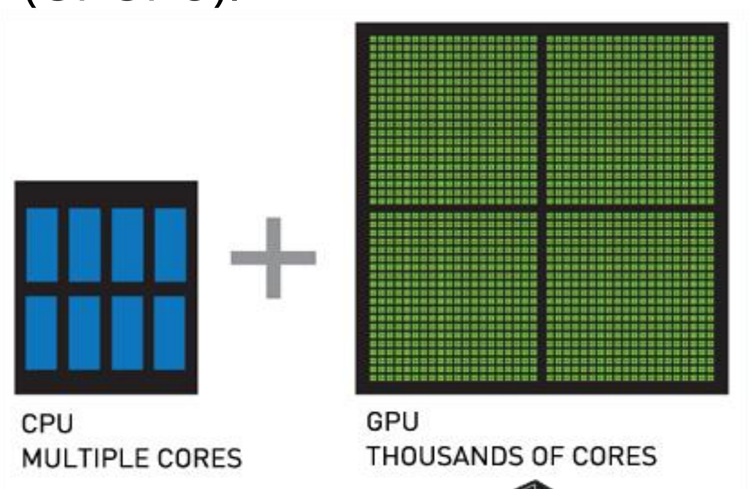
from xilinx.com

FPGA Development Process:

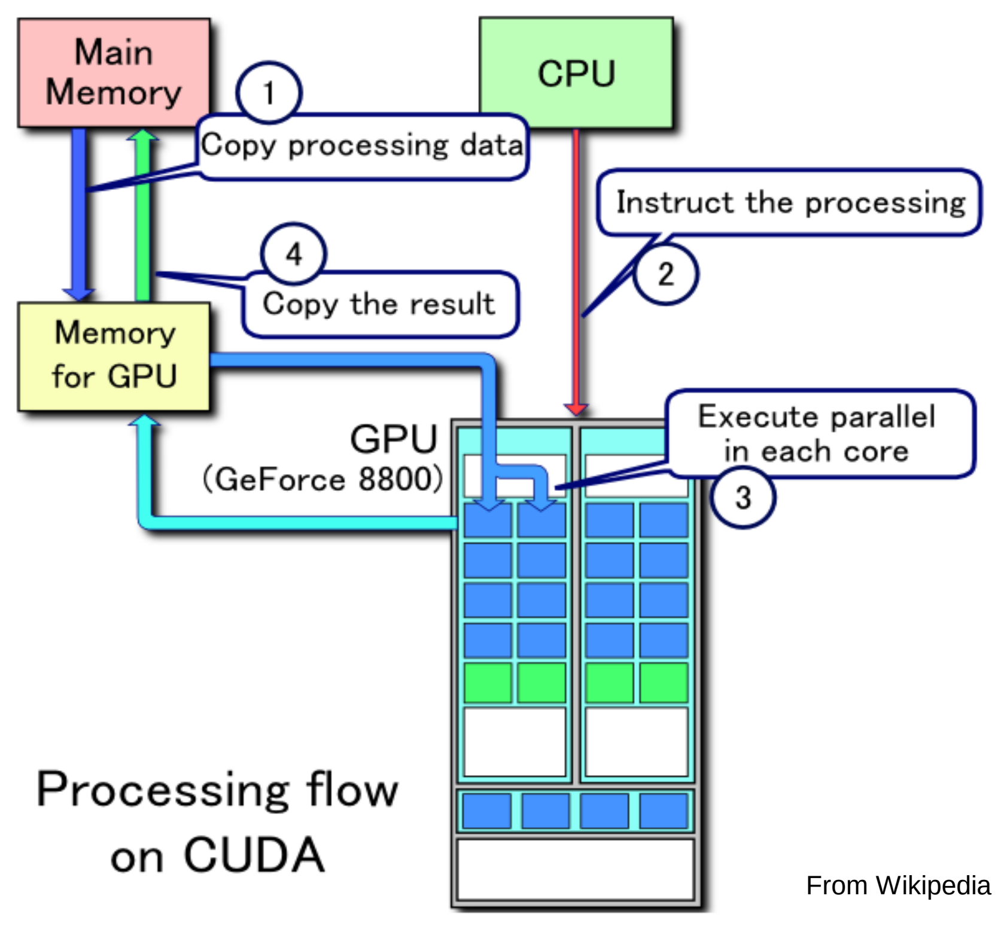

Floating-Point Algorithm (Matlab)
Fixed-Point Conversion
HDL Code Generation
HDL Verification & Co-Simulation (FIL)
Synthesis & Analysis (Generate Bitstream)
Target Device
Use Fixed-Point Designer to balance word-length against accuracy, sample-rate, footprint, critical path, quantization noise etc.
Select Verilog or VHDL
Examine Pipelines, break critical paths, Pipeline register insertion, Memory Mapping, Loop unrolling, Area optimization, etc.
Verify with HDL test-bench, through simulation between Matlab and HDL, use FPGA in the loop (FIL)
Place and Route

GPU Development

- GPUs are a very suitable for computationally-intensive applications as they can process hundreds or even thousands of pieces of data simultaneously. Now used for General Purpose computing (GPGPU).



Both from Nvidia.com



From Wikipedia

- GPUs are stream processors that can operate in parallel by running one kernel on many records in a stream at once. A *stream* is a set of records/instructions that require similar computation. Streams provide data parallelism. *Kernels* are the functions that are applied to each element in the stream.
- Nvidia's C-based CUDA (Compute Unified Device Architecture) in 2007 has facilitated huge growth in general-purpose programming on GPUs. OpenCL is a non-proprietary option.
- GPU acceleration is heavily used for accelerating simulations, but typically involves high power usage (more processors being used!)
- Mobile and Embedded GPUs – new generation of devices include Tegra (SoC) integrate GPUs with low-power ARM processors (as in Microsoft Surface). Forthcoming Nvidia 'Project Logan' are to be the first CUDA capable mobile/embedded hardware devices.

GPU Application in Development
GPU Acceleration of Support Vector Machine (SVM) and Gaussian Process (GP) Classifiers.

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) and Dstl