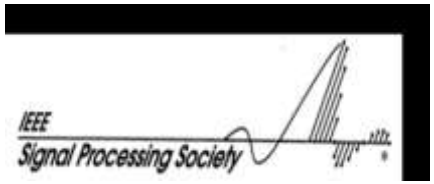
SSPD 2015

Sensor Signal Processing for Defence Conference



www.sspdconference.org

RCPE _ WiFi, password chiron1681

Roke

Part of the Chemring Group

Signal Processing in Real World Systems

SSPD Conference: Industrial Panel 10/09/2015

D A Johnson

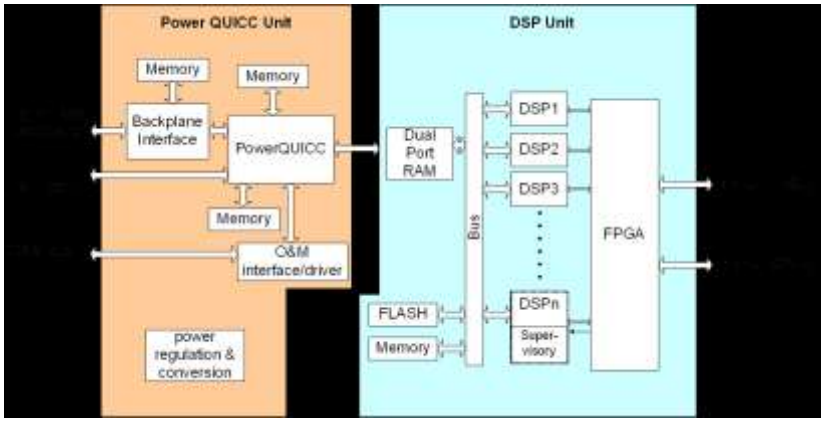
Roke

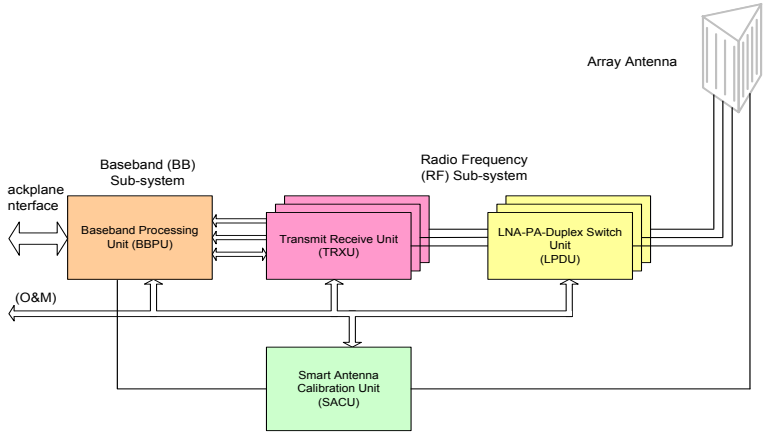
Intelligentinsight

www.roke.co.uk

Design Process

- Model Algorithms (eg Matlab in DP floating point)
- Estimate algorithm complexity (MOPS)
- Partition the system
- Either: Estimate type of hardware needed for solution
  - GPP
  - DSP
  - FPGA
- Or: Modify algorithm until complexity matches available hardware
  - Requirements may have to be relaxed if the hardware came before the algorithm
- The design process will then differ depending on target hardware





Design Process – Complexity Estimation

- Get A Rough guide: Calculate/Estimate
  - The number of numerical operations needed per second (NOPS)
    - A NOP can be Multiply or Add or MAC combined
    - Apply “Efficiency factor” conversion to get cycles for real code execution compared to maximum hardware compute capability
  - Input/Output sample rate (I/O bandwidth)
    - Dedicated I/O path for input samples
    - Inter-processor/inter-core interconnect bandwidth
    - Total Bus transfers <10% Bus bandwidth
  - Rough memory requirements
    - Largest blocks of data
    - Hand-waving from previous project code size
- Memory, latency and execution cycles can be traded depending on requirements

Design/Optimisation Process – GPP

- If Complexity estimation looks promising then proceed
  - (Note: the efficiency factor is low, typ. 3% for an X86 PC since attaining max hardware compute capability is limited by many factors)
- Convert high level description of algorithm to C/C++
  - Use libraries (eg IPP, FFTW etc) where possible
  - Start with most intensive processing function and Benchmark (measure clock cycles needed to execute)
  - Optimise code and refine efficiency factor
- Estimate number of cores required
- Design / partition software
- Write or auto-generate C/C++ code
- Validate each processing module against Matlab model
- Validate processing chain against Matlab model

Design/Optimisation Process – DSP

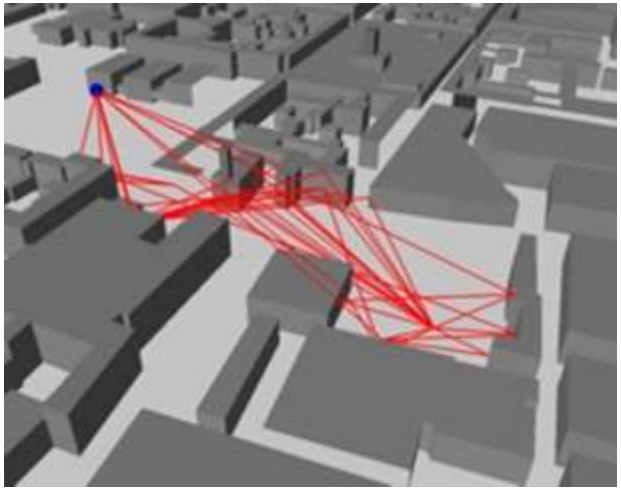
- DSP brings other complications on top of the GPP process:
  - Complexity estimation may factor in SIMD instructions (eg 4x16 bit MACs per cycle)
  - Custom accelerator block may be available (eg FFT, Viterbi etc)
  - Custom instructions (eg CMULT, ACS etc)
- 16 bit fixed point translation from floating point algorithm model
- Optimise only where necessary (code is less portable once intrinsics included)
- Using RTOS and task scheduling in multicore can ease development
- Eliminate dynamic memory allocations if possible
- Assembly coding is absolutely the last resort
  - Better to re-write C/C++ code so compiler can give a more efficient translation.
  - Assembly coding is too time consuming and difficult to debug
  - Improvement often marginal over compiler unless many custom instructions are used.

Design/Optimisation Process – FPGA

- Fixed point model Matlab
  - More effort at the design stage
  - Very useful for algorithms including feedback
  - VHDL design made easier
    - bit widths known
    - Test vectors from model give bit true agreement with implementation
- Explore implementation options before committing to VHDL
- Consider auto generation / high level synthesis (HLS)
  - HLS offerings from Matlab/Simulink, Xilinx, Altera
- Development timescales are likely to be more critical than implementation efficiency

Case Study 1: RESOLVE Geolocation

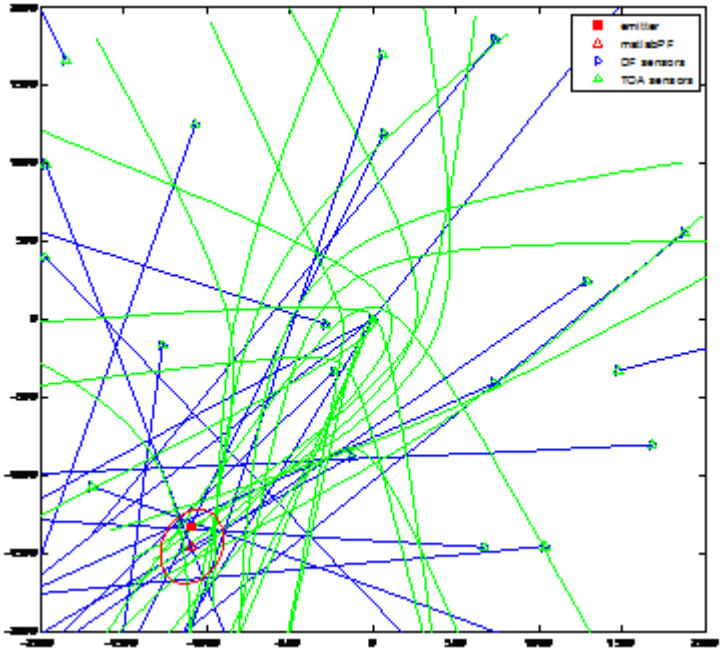
- The RESOLVE system is a wideband, man-pack sized, spectrum monitoring, signal identification /analysis and direction finding (DF) system.



- Butler Matrix DF accuracy effected by multipath
- N channel DF can resolve multipath but still not distinguish correct path
- Multiple RESOLVE sensors allow Position Fix (PF).

Case Study 1: RESOLVE Geolocation

- Discrete Probability Density Algorithm used for position fix with unreliable DF and TDOA.
- Academic paper proposes computation of probabilities at grid points and multiplication of probability values from different measurements at common grid points to get probability of emitter location.



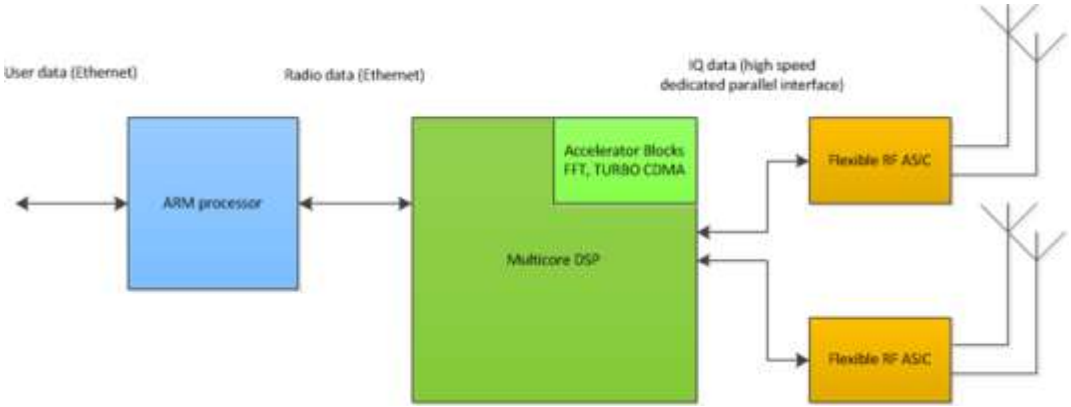
Case Study 1: RESOLVE Geolocation

- Real world problems:
  - Gaussian probability density assumed. In practice there are many wild measurements
  - Multiplication of a large number of probabilities gives numerical problems at single precision.
  - Sensors can be deployed in a cluster rather than distributed. Emitters close to cluster can disappear between grid points.
  - Computation over many fine grid points is not viable

- Real world solutions:
  - Compute Log probability with modified (non Gaussian) pdf.
  - Summing Log probabilities to combine measurements is ok numerically at single precision
  - Use non-uniform grid with higher density around sensor detects emitters close to sensors

Case Study 2: SmartLink Robust Comms

- SmartLink is Roke's rapidly deployable secure 3G comms system
- Modification of 3G receiver to be robust against jamming
- Transform excision approach to removing interference
- Modern DSP and RF ASIC Architecture



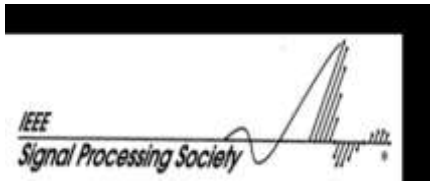
- Transform code written in C fails to keep up with the 3G sample rates
- Solution: Use the FFT accelerators and pipe-line processing

Conclusions and Statements for Discussion

- Design the algorithm for ease of implementation
- Dimension the hardware before building it
  OR
  Modify the algorithm so dimensioning shows it will fit the hardware available
- Assembly language development is only for when you get dimensioning wrong
- It is essential to make use of RTOS, function Libraries and hardware accelerators
- Development timescales and budget are the most difficult thing to meet

SSPD 2015

Sensor Signal Processing for Defence Conference



www.sspdconference.org

RCPE _ WiFi, password chiron1681