
QinetiQ

Adaptive beamforming using QR in FPGA

Richard Walke, Real-time System Lab
Advanced Processing Centre
S&E Division

QinetiQ

Contents

1. Architecture of adaptive beamformer
2. FPGA components
   - Digital receiver
   - QR processor – for adaptive weight calculation
3. Design methodology
4. Demonstration overview
5. Conclusions

QinetiQ

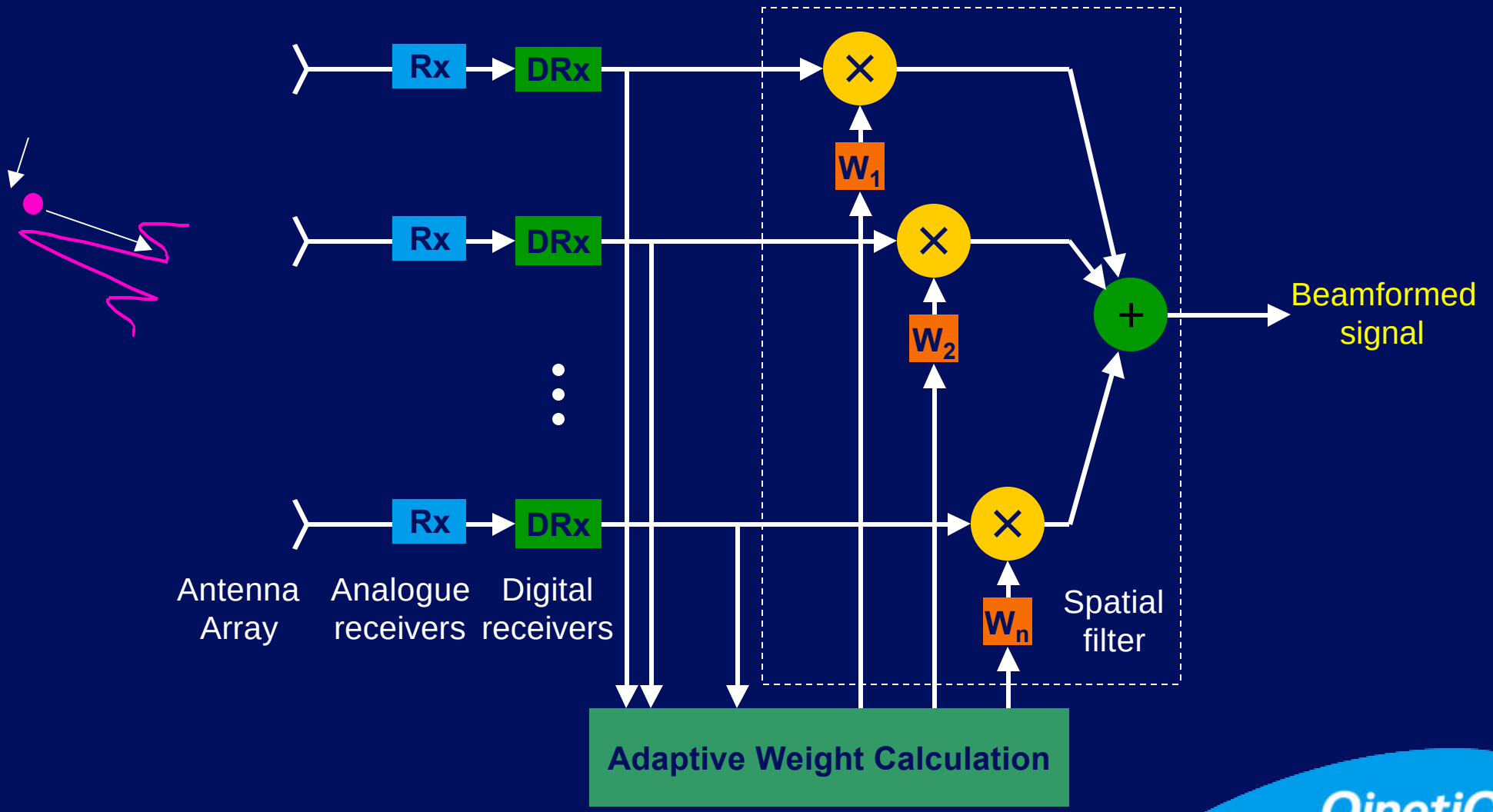
Section 1

Architecture of adaptive beamformer

QinetiQ

Architecture of adaptive beamformer

Adaptive beamformer



QinetiQ

Section 2

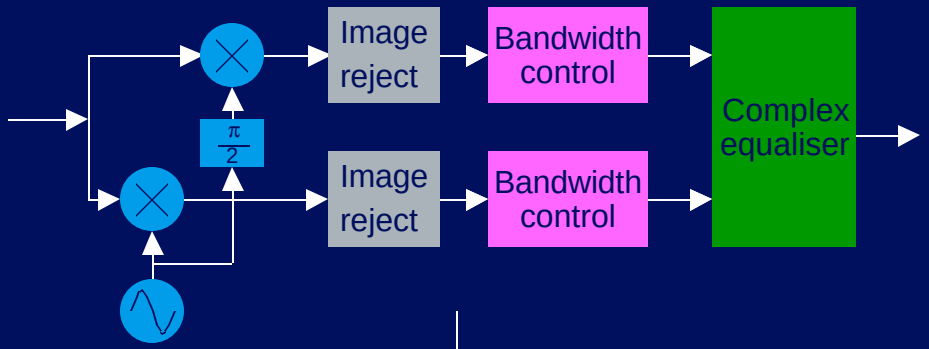
FPGA components

QinetiQ

FPGA Components

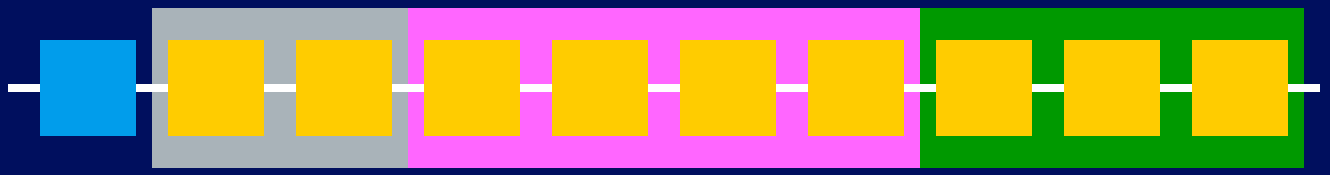
Software configurable FIR

Radar receiver specification



Runtime configuration parameters

Array of programmable ‘tap-processors’



QinetiQ

FPGA Components

Software configurable FIR

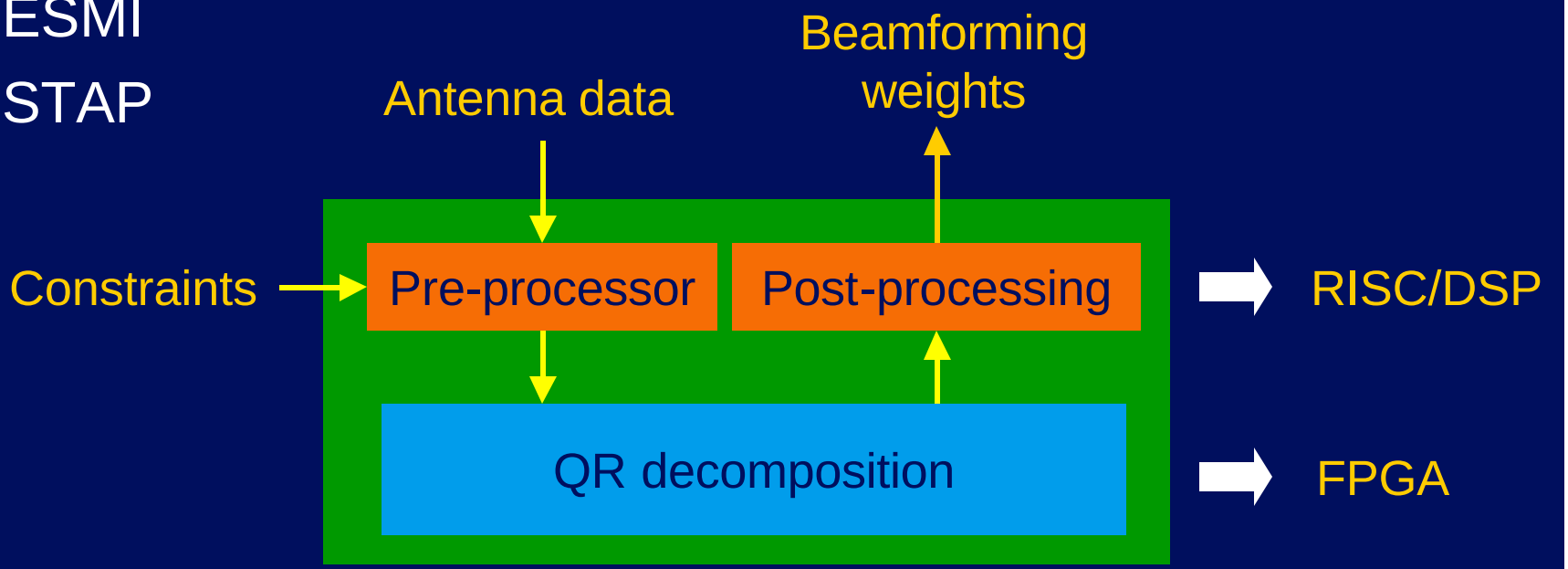
- Software programmable parameters include:
  - filter length
  - decimation ratio
  - complex/real arithmetic
  - number of channels
  - time varying filtering (inter & intra-pulse)
- Performance 20-30 GOPS on XC2V6000-5
  - 100 GOPS on Virtex2 Pro (2003)

QinetiQ

FPGA Components

Weight calculation using QR

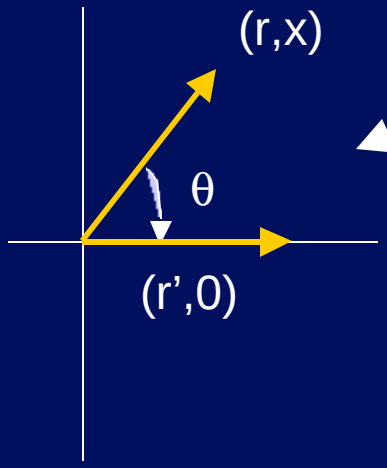
- Building block for a range of adaptive algorithms
  - Sample matrix inversion (SMI)
  - Soft constraints
  - ESMI
  - STAP



FPGA Components

QR decomposition

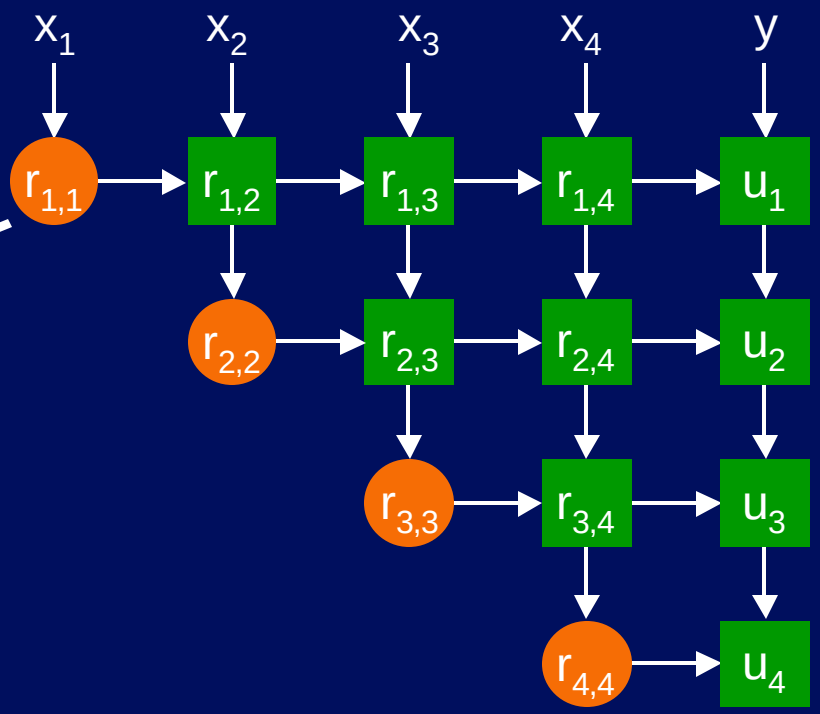
Vectorise cell:
11 real floating-point operations



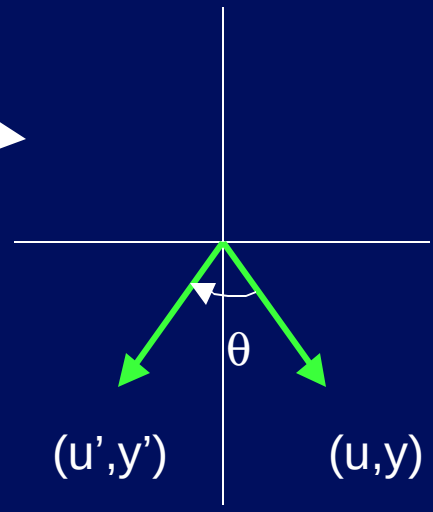
$$\cos\theta = \frac{r}{\sqrt{r^2+x^2}}$$

$$\sin\theta = \frac{x}{\sqrt{r^2+x^2}}$$

Input data



Rotate cell:
16 real floating-point operations



$$\begin{bmatrix} u' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} u \\ y \end{bmatrix}$$

For SMI: Rw=u

QinetiQ

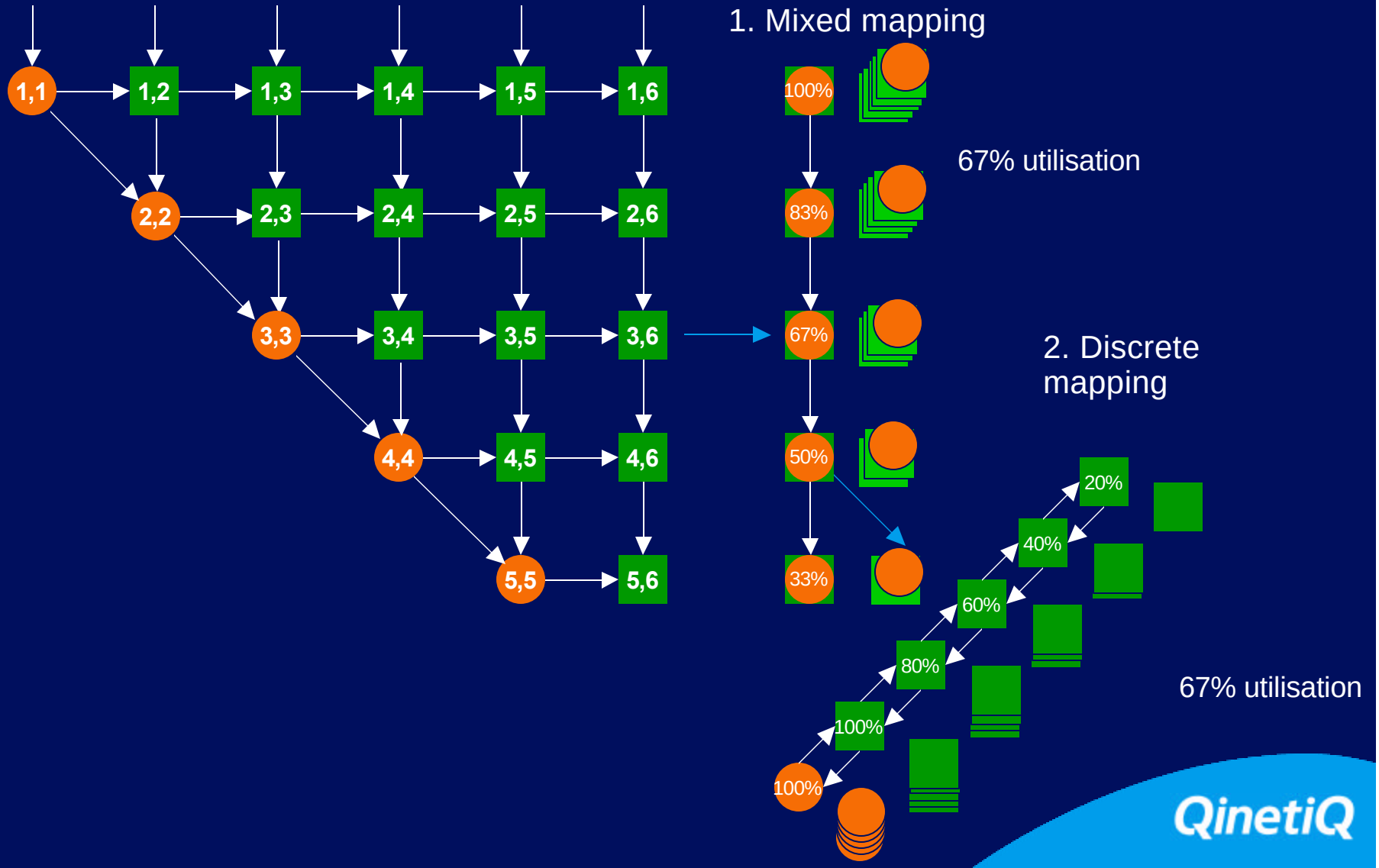
FPGA Components

Features of QR

- Good numerical properties. Arithmetic choices:
  - CORDIC: shift-add
  - Fixed-point: multiply-add
  - Floating-point: Higher dynamic range, allows algorithms with fewer operations & lower wordlength. Smallest!
- Highly parallel (Givens rotations)
  - Suits FPGA
  - Need to reduce parallelism for many applications!

FPGA Components

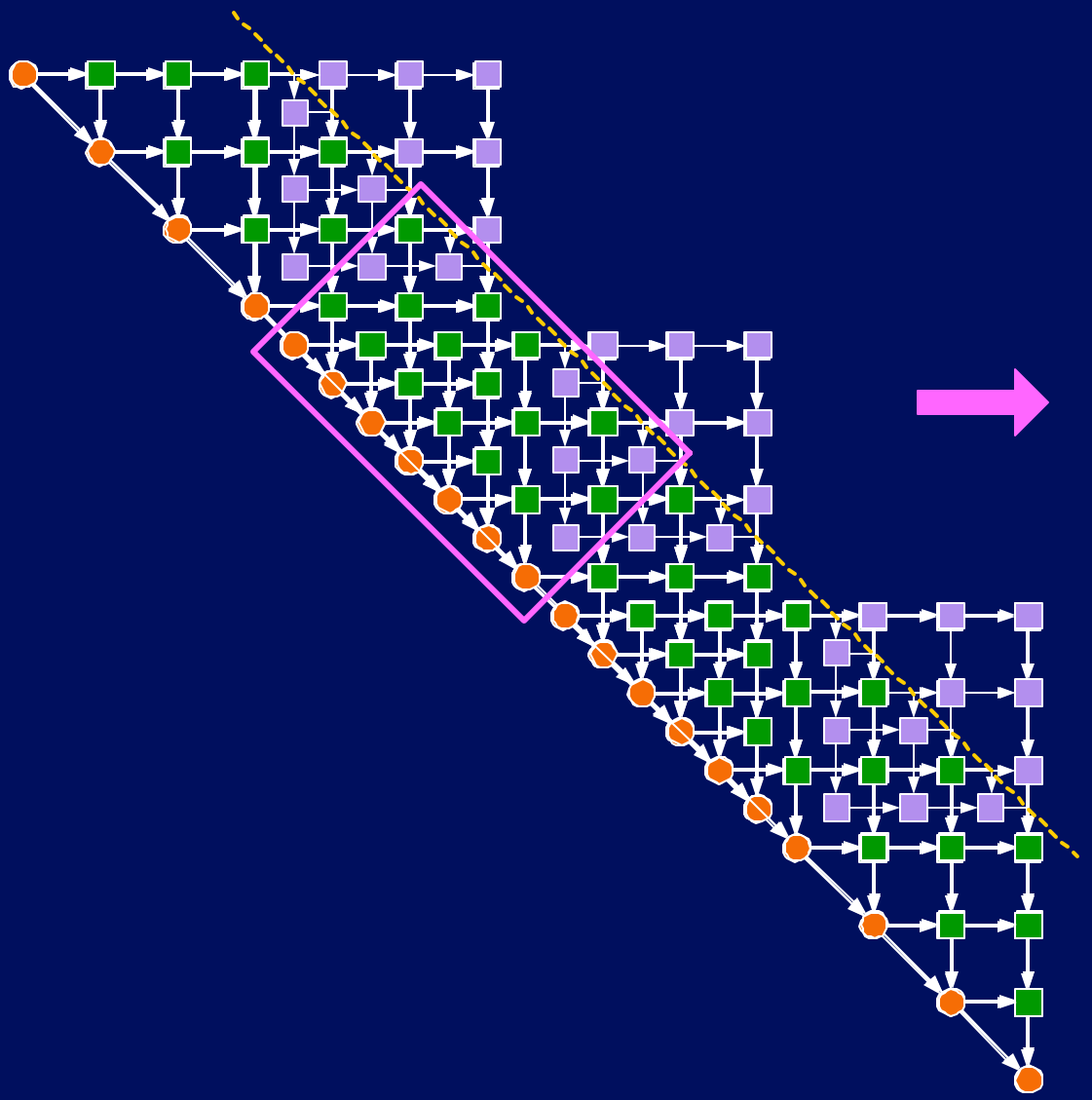
Obtaining lower-levels of parallelism

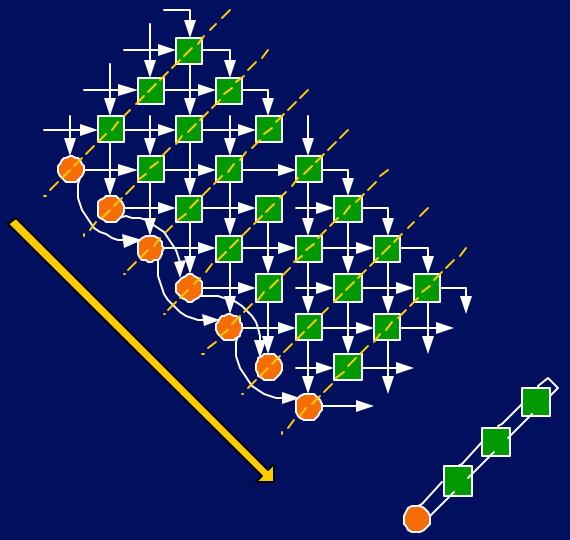


QinetiQ

FPGA Components

Novel mapping of QR



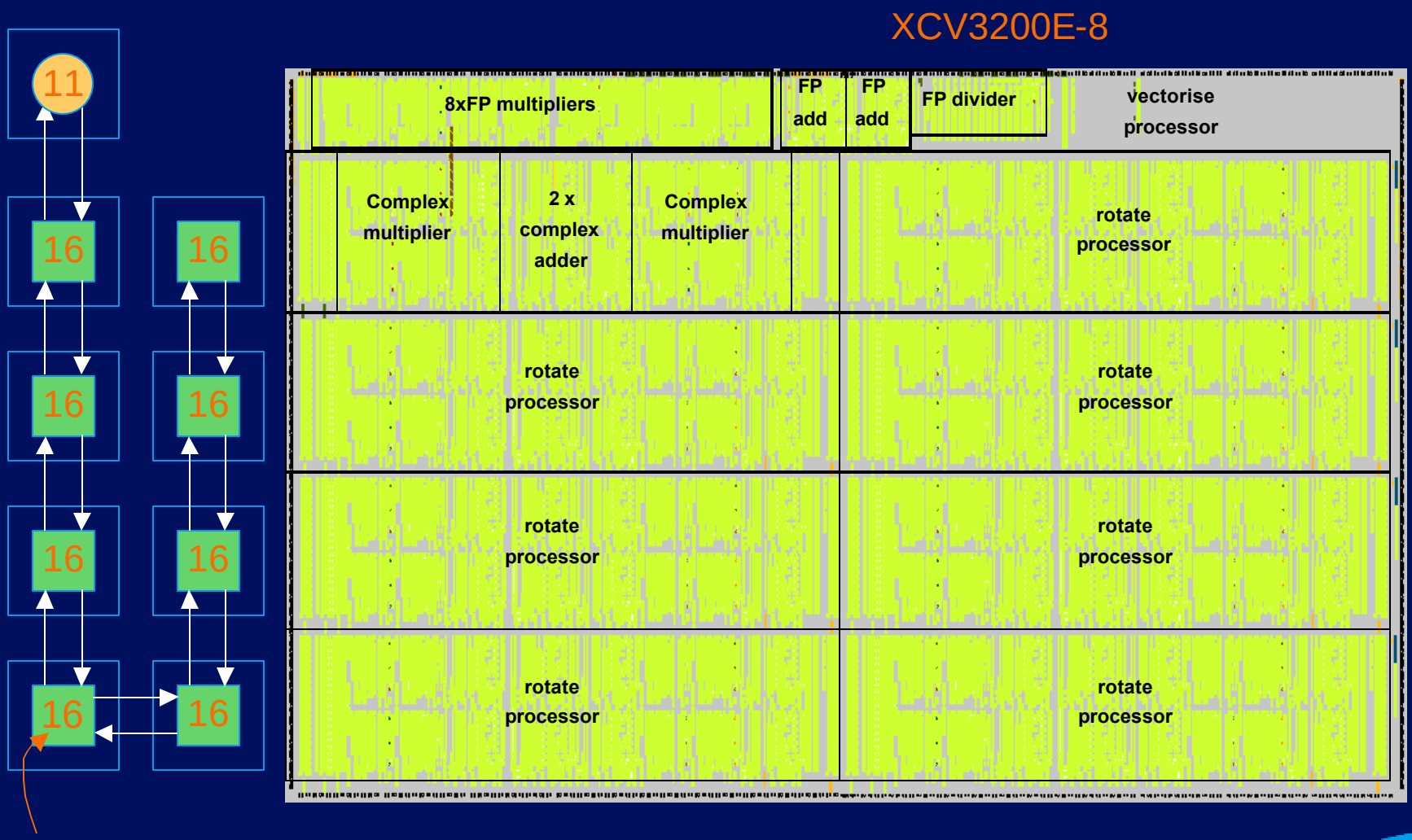


Linear systolic array

QinetiQ

FPGA Components

FPGA implementation



139 14-bit FP operators @ 160MHz = 22 GFLOPS

QinetiQ

FPGA Components

QR processor - main features

Size	Mantissa wordlength	Clock	Utilisation (XC2V6000)				Operations	Power
1 Boundary 3 Internal	14-bit mantissa	101MHz[5]	Mults	32	22%	23%	6 GFLOPS	2.24W[4]
			Rams[3]	34	23%			
			LUTS	15K	22%			
			FFs	16K	23%			
1 Boundary 12 Internal	14-bit mantissa	100MHz[2]	82%[2]				20.3 GFLOPS	8W[6]
1 Boundary 9 Internal	17-bit mantissa	97MHz	74%				15 GFLOPS	7W[6]
Pentium™ 4 2GHz[1]							4 GFLOPS	70W

x50

1 Estimated (based on data from Richard Linderman)
2 Estimated (design too large for PC)
3 Also depends upon number of inputs
4 Obtained via Xpower
5 For XC2V6000-5
6 Extrapolated

QinetiQ

Section 3

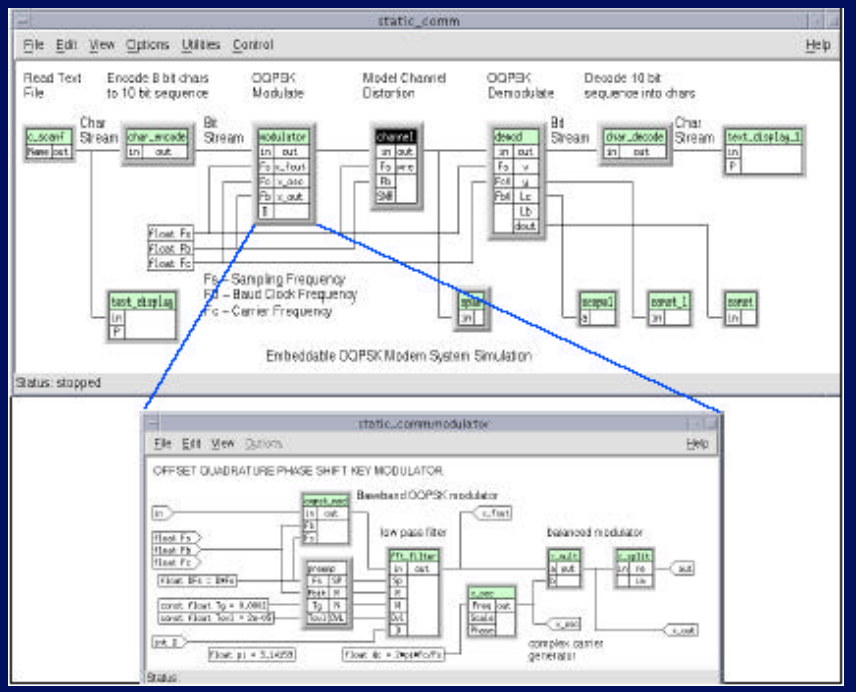
Heterogeneous design methodology

QinetiQ

Heterogeneous design methodology

GEDAE™

- Graphically specify system

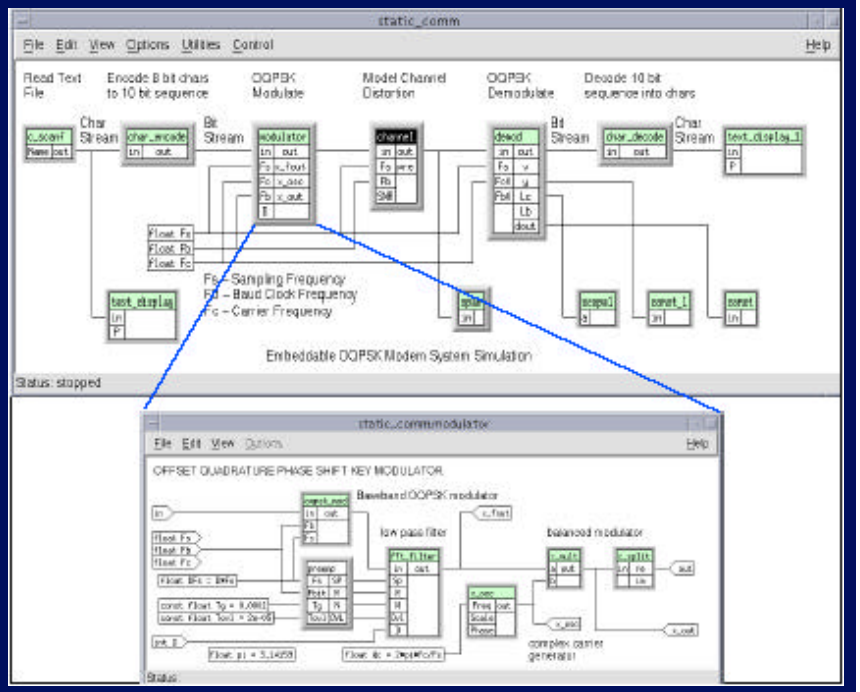


QinetiQ

Heterogeneous design methodology

GEDAE™

- Graphically specify system
  - primitive functions in ‘c’

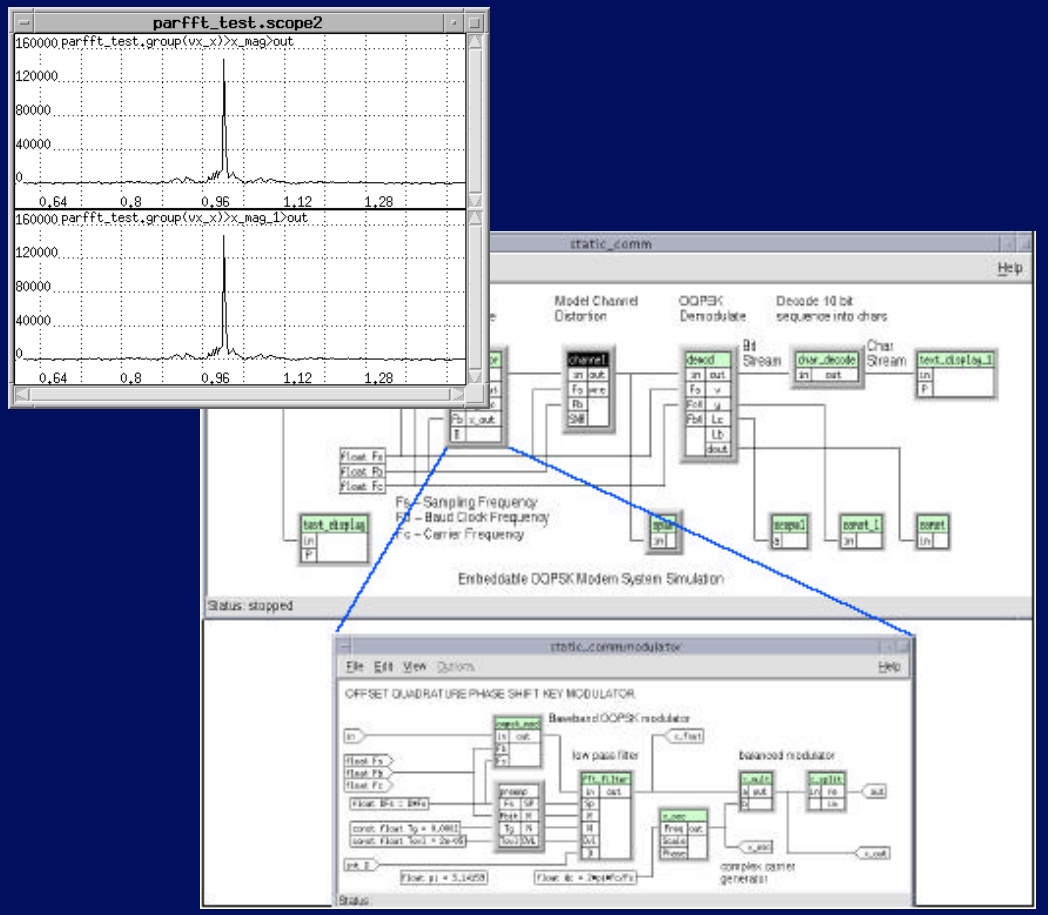


QinetiQ

Heterogeneous design methodology

GEDAE™

- Graphically specify system
  - primitive functions in ‘c’
  - executable specification

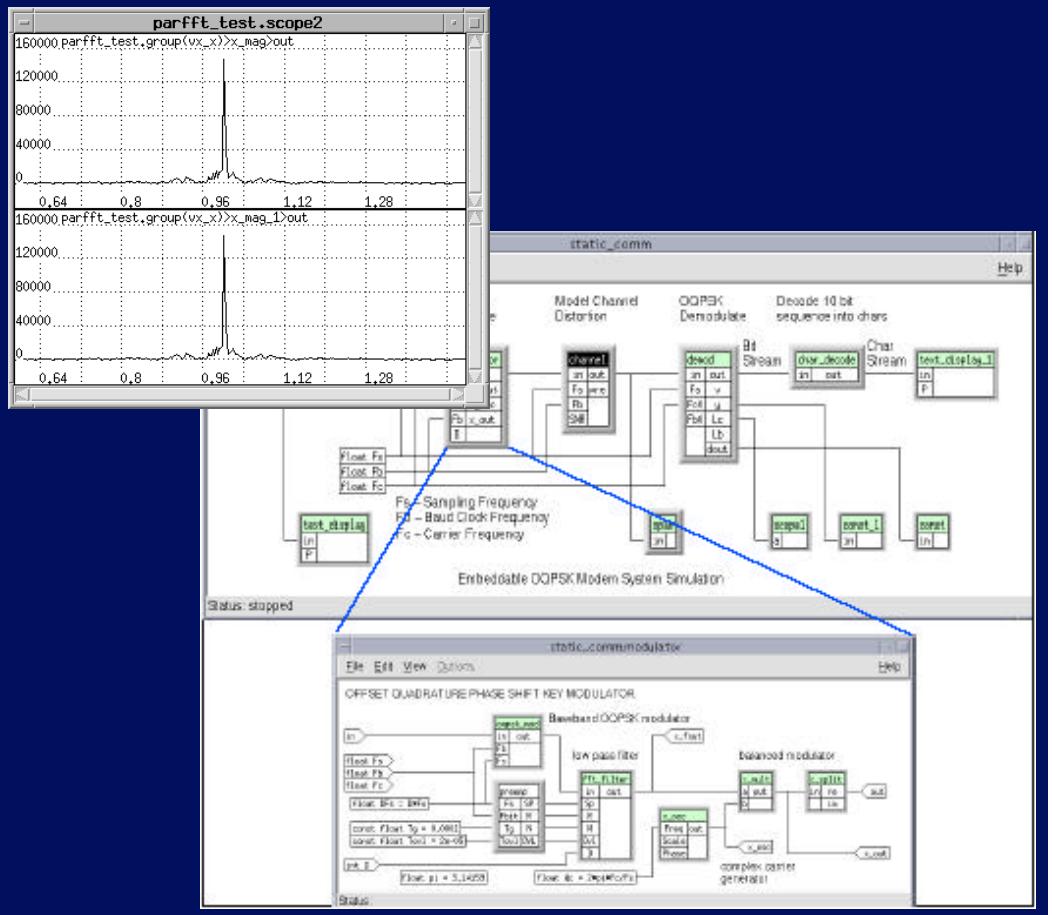


QinetiQ

Heterogeneous design methodology

GEDAE™

- Graphically specify system
  - primitive functions in ‘c’
  - executable specification
- Auto-code generation
  - parallel programme constructed by GEDAE

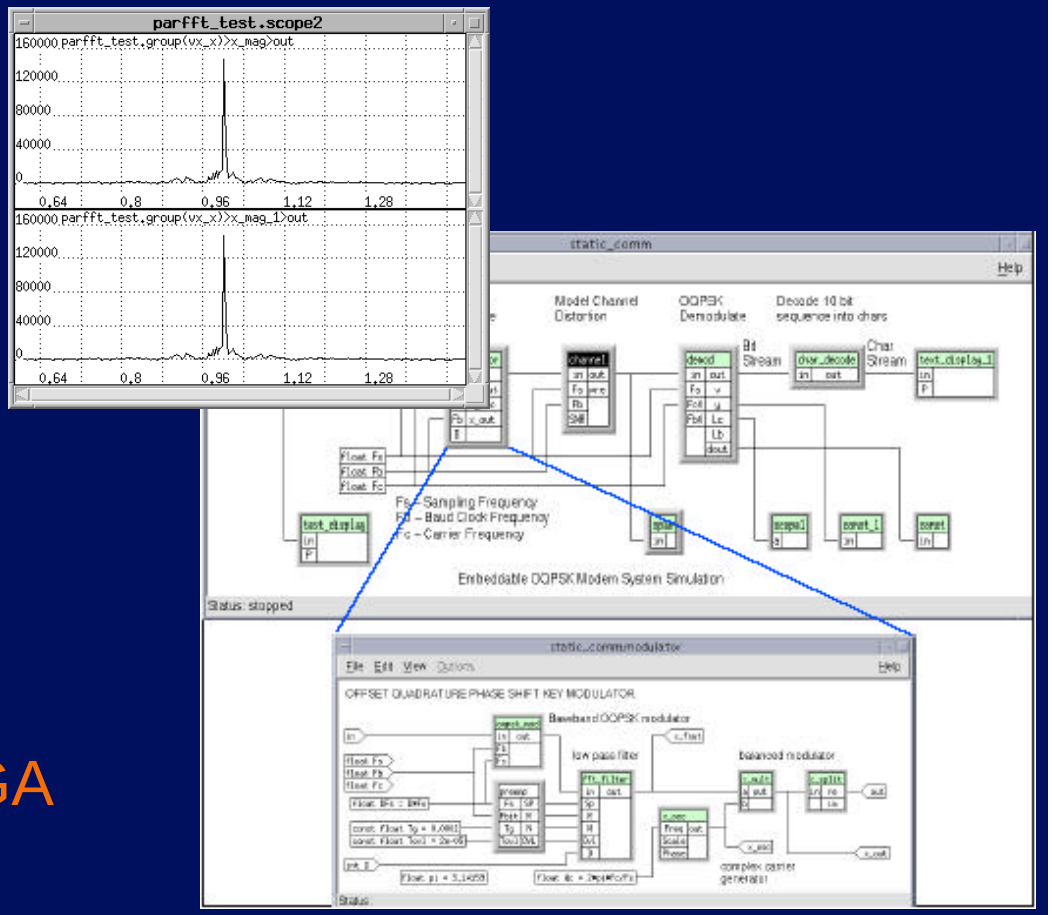


QinetiQ

Heterogeneous design methodology

GEDAE™

- Graphically specify system
  - primitive functions in ‘c’
  - executable specification
- Auto-code generation
  - parallel programme constructed by GEDAE
- Currently no support for FPGA
  - highly compatible model

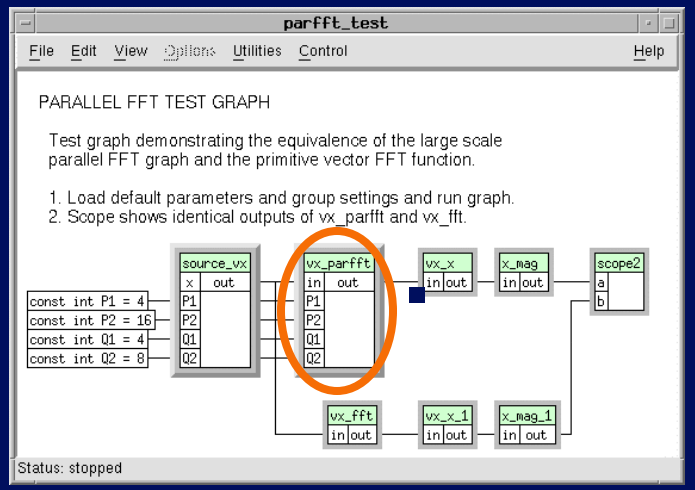


QinetiQ

Heterogeneous design methodology

Core based methodology

- Cores used for key functions
  - FFT, QR, FIR filter ...
  - Build in parallelism (manually)
  - Parameterised

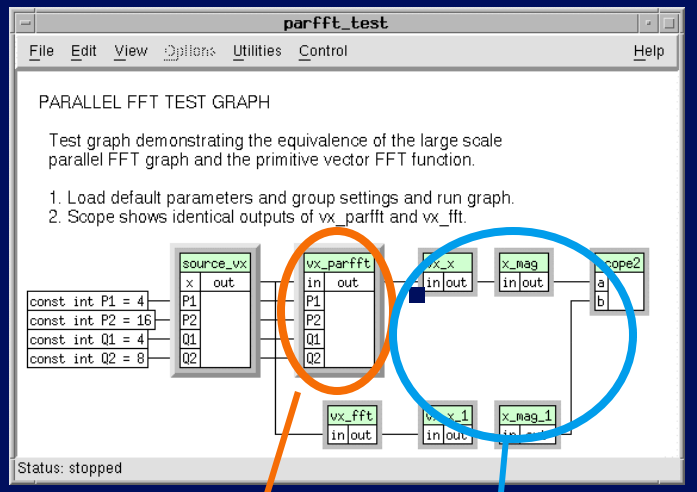


QinetiQ

Heterogeneous design methodology

Core based methodology

- Cores used for key functions
  - FFT, QR, FIR filter ...
  - Build in parallelism (manually)
  - Parameterised
- Automatically generated system
  - communications inserted



FFT core from library

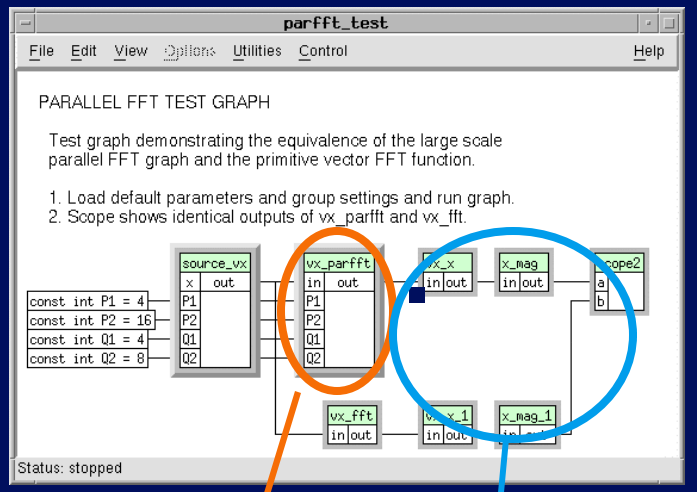
FPGA → Processor

QinetiQ

Heterogeneous design methodology

Core based methodology

- Cores used for key functions
  - FFT, QR, FIR filter ...
  - Build in parallelism (manually)
  - Parameterised
- Automatically generated system
  - communications inserted
- Architectural exploration
  - Compaan gives Matlab NLP to VHDL
  - RTL output in future version



FFT core from library

FPGA → Processor

QinetiQ

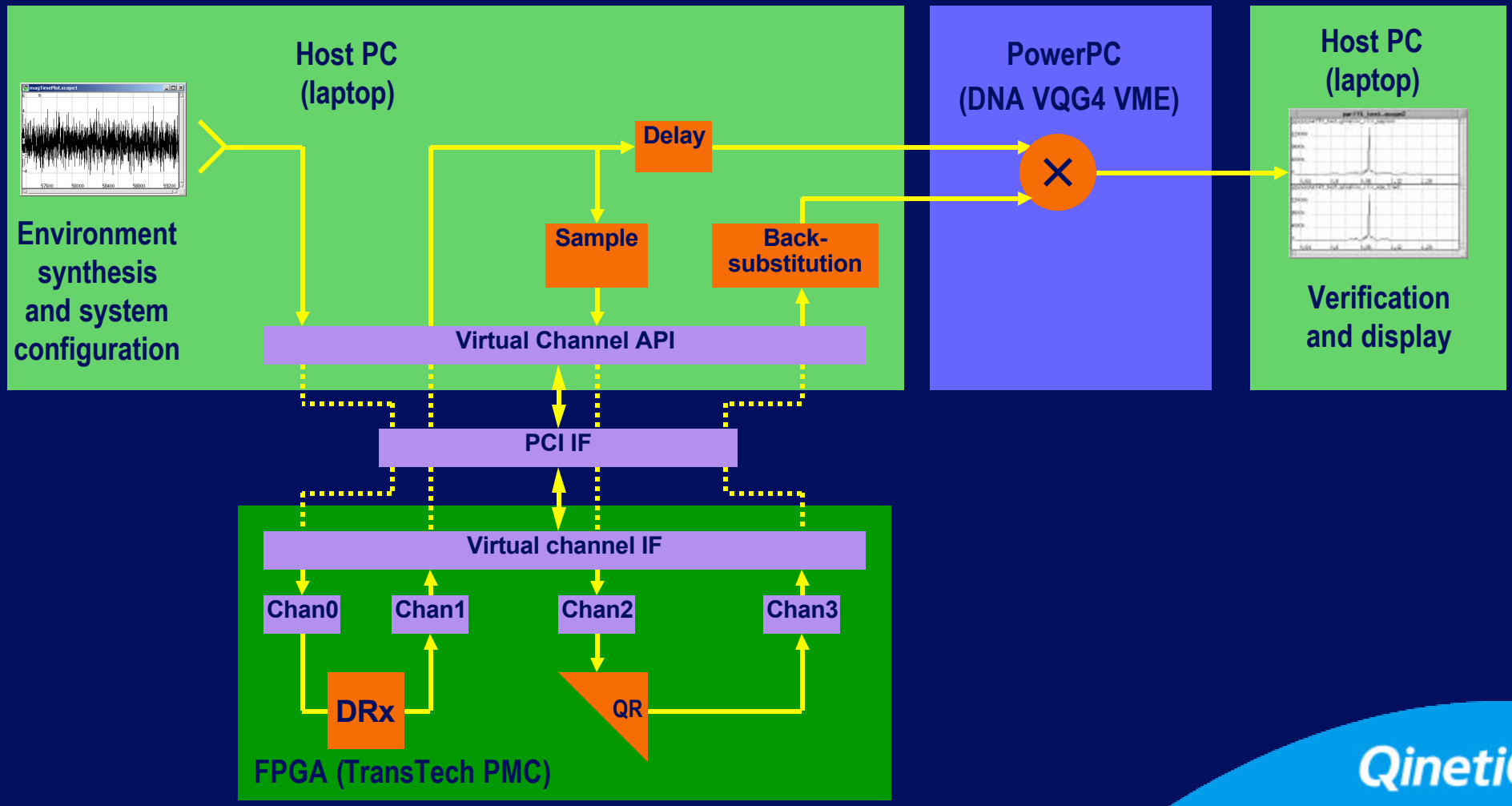
Section 4

Adaptive beamformer demonstration overview

QinetiQ

Demonstration overview

System mapping



QinetiQ

Conclusion

- FPGAs
  - Performance dependent upon level of optimisation
  - Floating-point is realistic
  - 10x compute improvement
  - 5 - 20x power improvement
- Design is main issue
  - Hardware design: High levels of parallelism required
  - Core-based design approaches offer interim solution
  - Architectural synthesis tools are emerging

Acknowledgements

- The project to develop a core-based design methodology is a collaboration between:
  - QinetiQ Ltd
    - poc: rlwalke@qinetiq.com
  - BAE SYSTEMS ATC, Gt Baddow
    - poc: ian.alston@baesystems.com
- Contributions have been made by John McAllister under contract with the Queen's University of Belfast.
- This work was sponsored by the United Kingdom Ministry of Defence Corporate Research Programme.

QinetiQ


QinetiQ