PACT XPP Technologies

PACT XPP Technologies

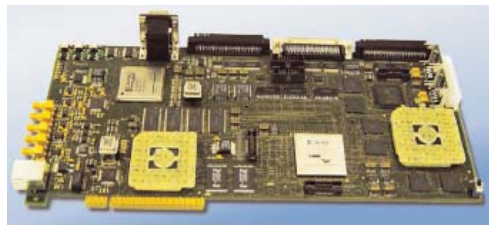
- Fabless semiconductor company
- Located in Munich, Germany and office in Los Gatos, CA
- The pioneer in coarse grained **Dynamic Re-configurable Processor Platforms**
- High Performance DSP solutions for Media Servers, Multimedia, SDR
- 26 patents granted on reconfigurable computing, including base patents. 32 more filed.

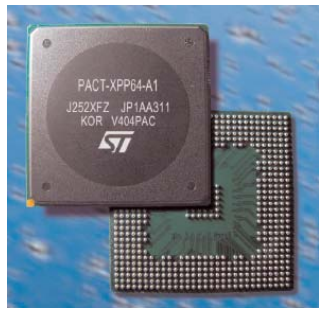
Milestones

1994-1999 10 base patents on reconfigurable processor technology filed

1999 PACT XPP Technologies founded

2003 successfully released world first commercial Dynamically Re-configurable Processor XPP64-A





2004 Optimized XPP-IIb IP

2005 Video Codec Kernels available

2006 world-first full ANSI-C compatible compiler for Dynamically Re-configurable Processors available

---

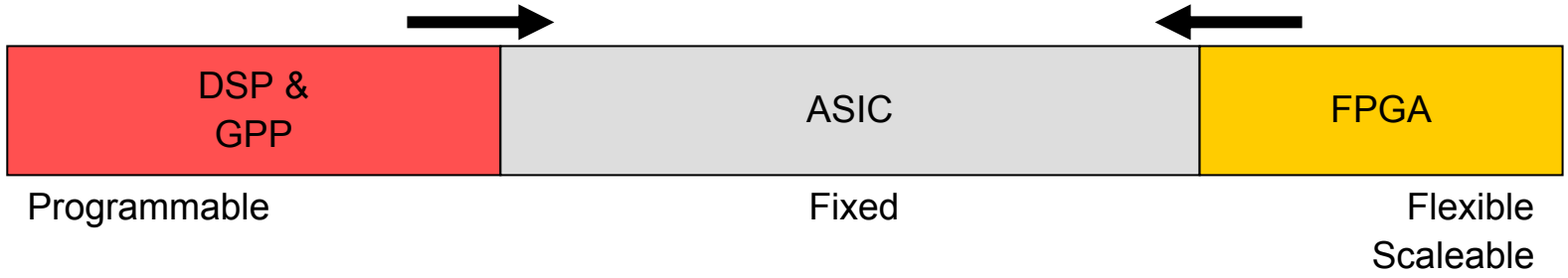
2007 XPP-III IP release

'Chip-Project' start

XPP-III market introduction

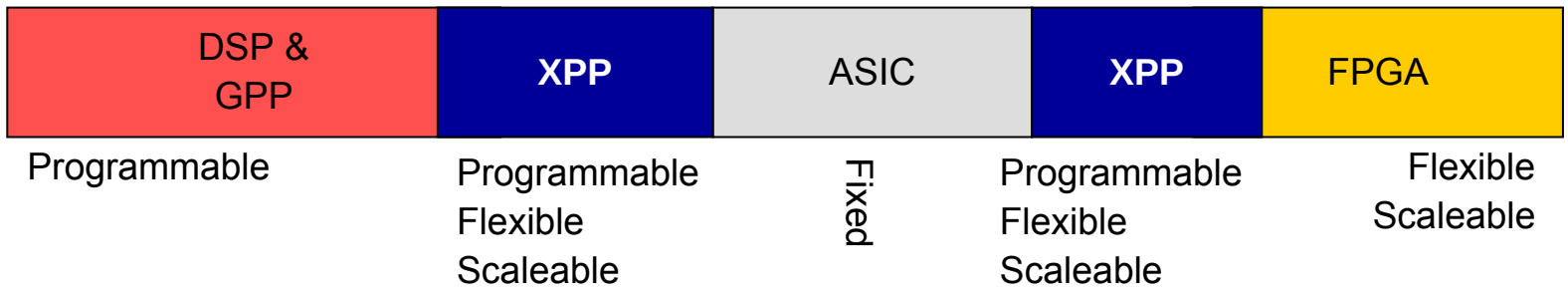
Market Positioning

Processors and FPGAs gathering market shares from fixed function devices



XPP-III offers all benefits derived from Programmability, Flexibility & Scalability

- high performance markets from processors
- high volume and development time critical markets from FPGAs



XPP-III Summary

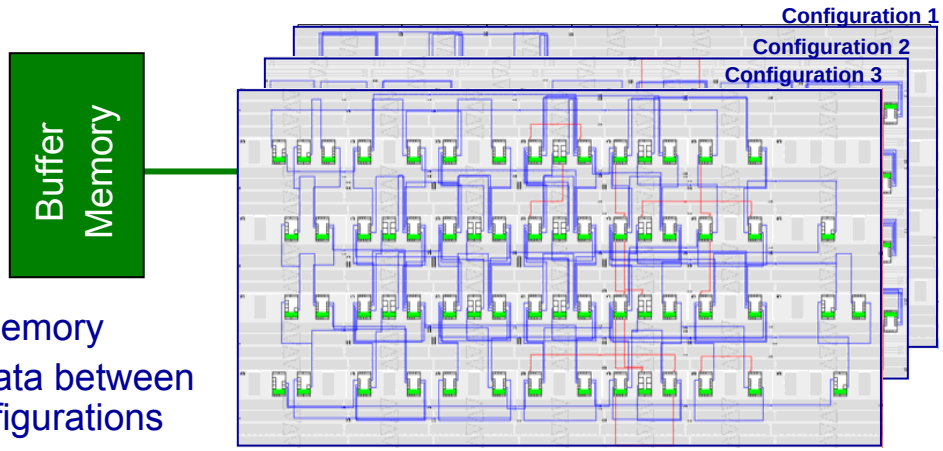
XPP-III Architecture Overview: Array

Processing model

- Data and control flow-graph
- Automatic data synchronization

Dynamic reconfiguration

- Fast configuration sequencing
- Local buffer memories / Fifos



Buffer Memory
stores data between configurations

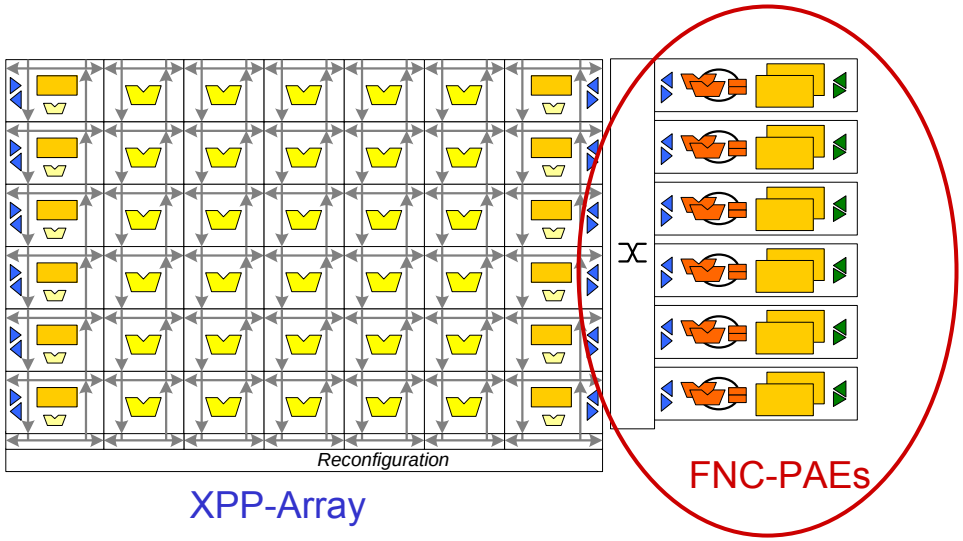
High efficiency and programmability

- Optimised for high bandwidth data streaming
- Small control overhead, no instruction fetches
- Programming: native flow-graph description and vectorizing C-compiler
- Simulator and graphical dataflow debugger

XPP-III Architecture Overview: FNC-PAEs

Function PAEs

- Sequential execution with 8 parallel ALUs each
- One cycle multi-condition evaluation and branching
- Local I-cache and TCM/D-cache
- Tightly coupled to the array via X-bar and data streaming channels
- 16-bit fixed-point core with 32-bit extensions and 32-bit addressing



New:
Multicore arrangement of small VLIW-like processor cores

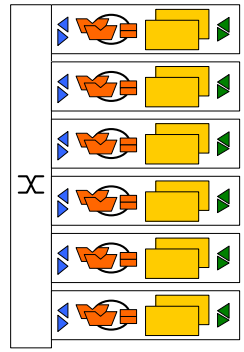
High efficiency and programmability

- Optimised for control-flow based algorithms
- C++ Compiler (GNU), Libraries and APIs
- Development suite incl. Simulator

XPP Application Space

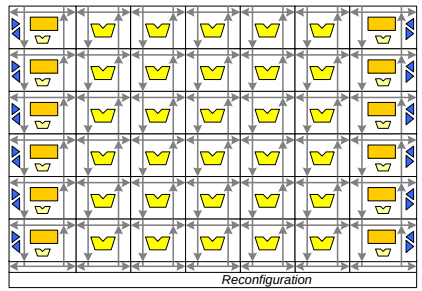
Function-PAEs

- Controlflow dominated processes
- Protocol Handler, Parsing, Crypto, Encoder/decoder, Huffman
- Legacy Software
- RTOS
- Housekeeping tasks:
  - DMA setup
  - Configuration sequencing
  - Interrupt handling
  - Peripherals setup



XPP-array

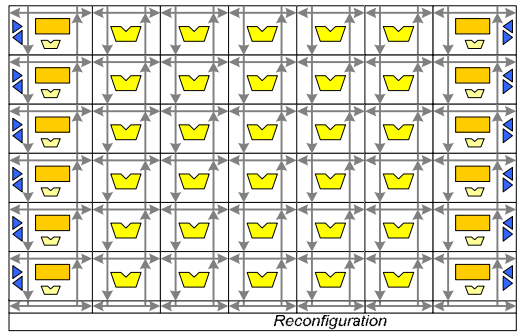
- Dataflow dominated processes
- Data streams processing
- SDR: FFTs, Filter, Modulators/demodulators, CDMA, Header extraction ..
- Media: Raw Pixel processing, DCT/Wavelet Transforms, Audio, Filter



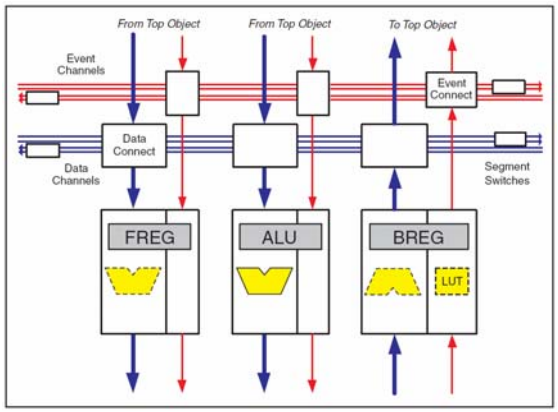
XPP-III Array Details

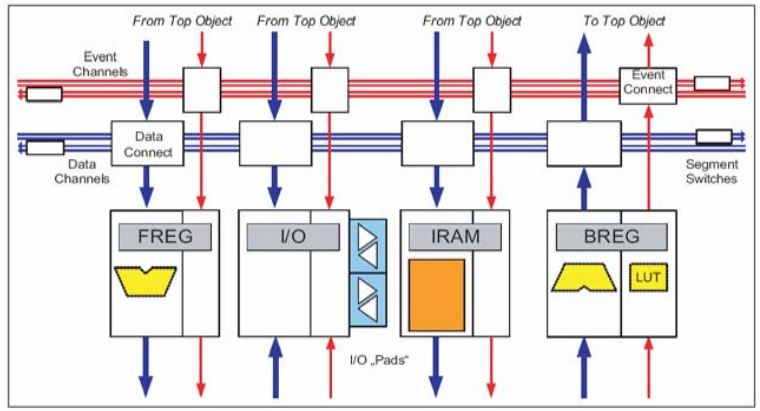
XPP Array

- Scalable Array of (columns/ rows) ALU-PAEs
- 2 columns of RAM-PAEs
- IO implemented in each RAM-PAE
- 16, 24, 32 bit fixed point



RAM PAEs - ALU PAEs - - RAM PAEs





ALU-PAEs

- Horizontal routing resources
- DSP-like center ALU (incl. MUL, double result)
- Side ALUs (incl. Dataflow Opcodes & Accumulator, barrel shift and vertical routing resources)

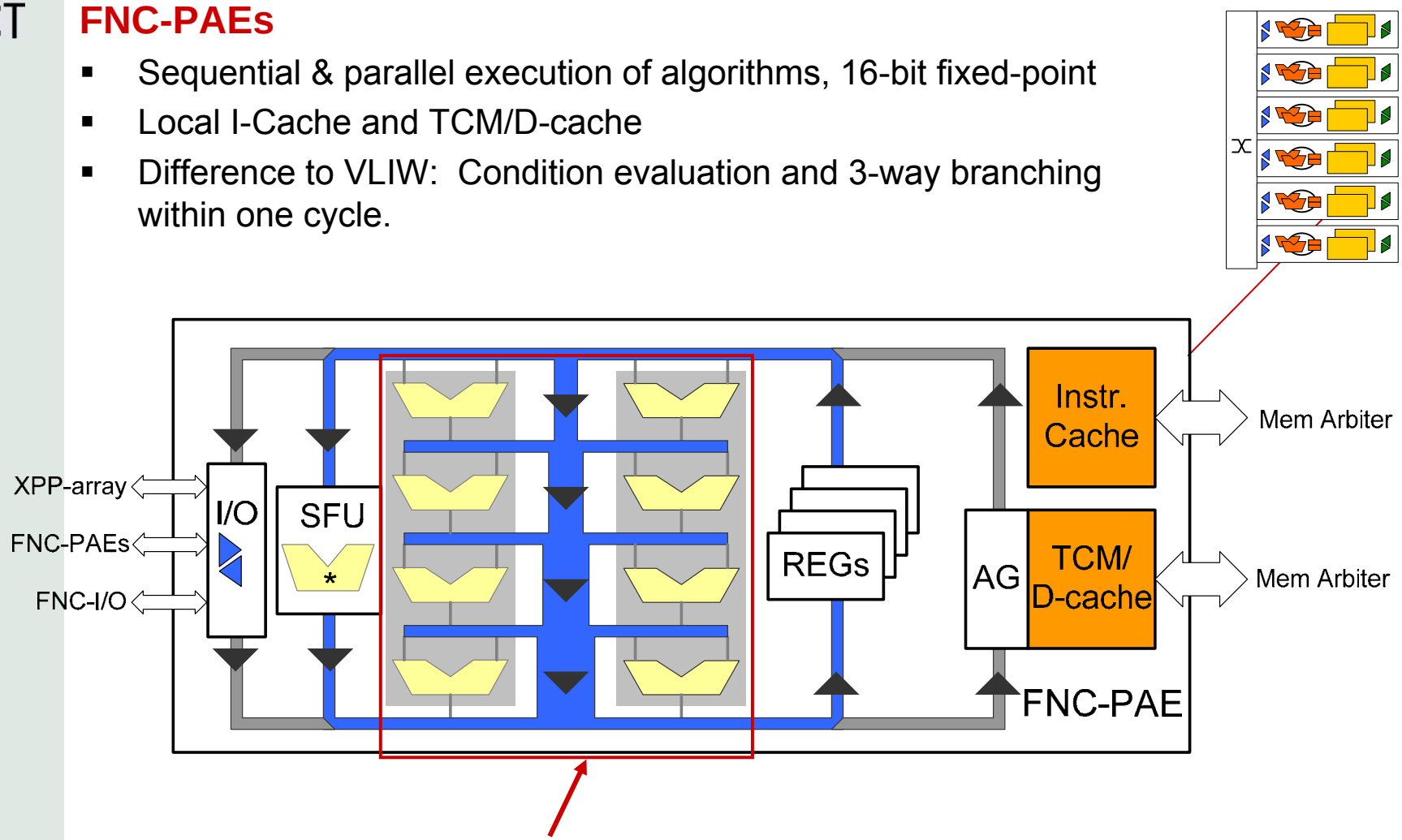
RAM-PAEs

- Horizontal routing resources
- SRAM memory (typ. 512 words)
- IO channels
- Side ALUs (same as in ALU-PAE, e.g. used for address generation)

FNC-PAE: Function Processing Element

FNC-PAEs

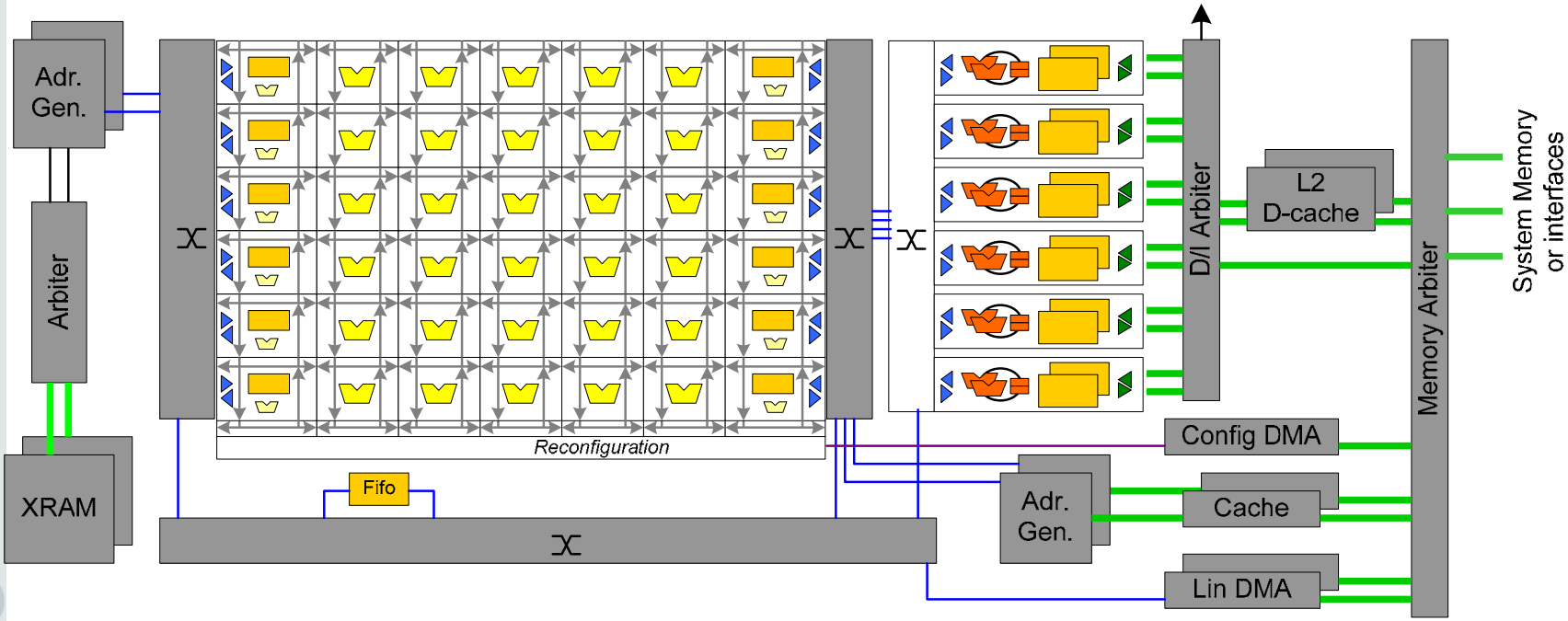
- Sequential & parallel execution of algorithms, 16-bit fixed-point
- Local I-Cache and TCM/D-cache
- Difference to VLIW: Condition evaluation and 3-way branching within one cycle.



- Asynchronous datapath, without pipeline stages
- Each ALU gets operands from register file and all ALU outputs above (simultaneous)
- Each ALU capable to write results into register file (simultaneous)

Modular and Scalable System and Communication Framework

- Crossbars for streaming channels
- Local Memories (XRAM), Fifos. L2 data chache
- 4D-Address Generators + R/W cache for typical Video access patterns
- Linear DMA and Configuration DMA
- Interrupt handler with programmable priorities
- Fully pipelined memory access arbiters with programmable priorities
- Standardized 16-bit streaming (blue) and 64-bit memory channels (green)



XPP-III IP scalability & parameters

- Array size:
  - 3x3 ALU-PAEs … any rectangular array,
  - RAM-PAEs = 2x number of rows
  - Word width: 16, 24, 32-bit
- FNC-PAEs: 0 .. 16 (can run in dedicated clock domain)
- Address Generators & caches: 0 .. 4
  - Read: 1k .. 4k x 256
  - Write: 0.5k .. 2k x 256
- Linear DMA: 0 .. 4
- Config DMA: 1 (fixed)
- XRAM: 0 .. 2
  - 2k .. 8k x 64-bit
- Fifos: 1 .. N, (0.5k ..4k x 16)
- L2-Data-Cache: 0 .. 2, (4k .. 8k x 64)
- X-Bars: up to 31 16-bit ports.
- Memory Arbiter: 1
  - Requester ports: 1 .. 32, 64-bit
  - 64-bit Memory channels: 1 .. 4

→ IP building blocks can easily be parameterized to processing needs
From stream processor or coprocessor to self-contained DSPs

Array Performance Data *(scaled down to 100MHz)*

FFT (@100 MHz, XPP array)

- 2x complex streaming I/O, double internal results

kFFTs / sec	256	1024
XPP40	195	32
XPP60	390	48

FFT size

2D-DCT (@100 MHz, XPP array)

- 16-bit streaming I/O with 4D Addr. Generator, 8x8 macroblocks

kDCTs / sec	8x8
XPP40	3120
XPP60	4680

Wavelet with partial reconfiguration (@100 MHz, XPP-II array)

- *Floating point* implementation of a 5/3 wavelet decomp. on the fixed-point array
- 4 .. 6 ALU PAEs per FP operation

Time[ms]	512x512	1024x1024
XPP64	5263	20972

image size

FNC-PAE Performance Data (scaled down to 100MHz)

AES (@100 MHz)

- 128 bit key encode or decode
- Scales with number of FNCs involved.

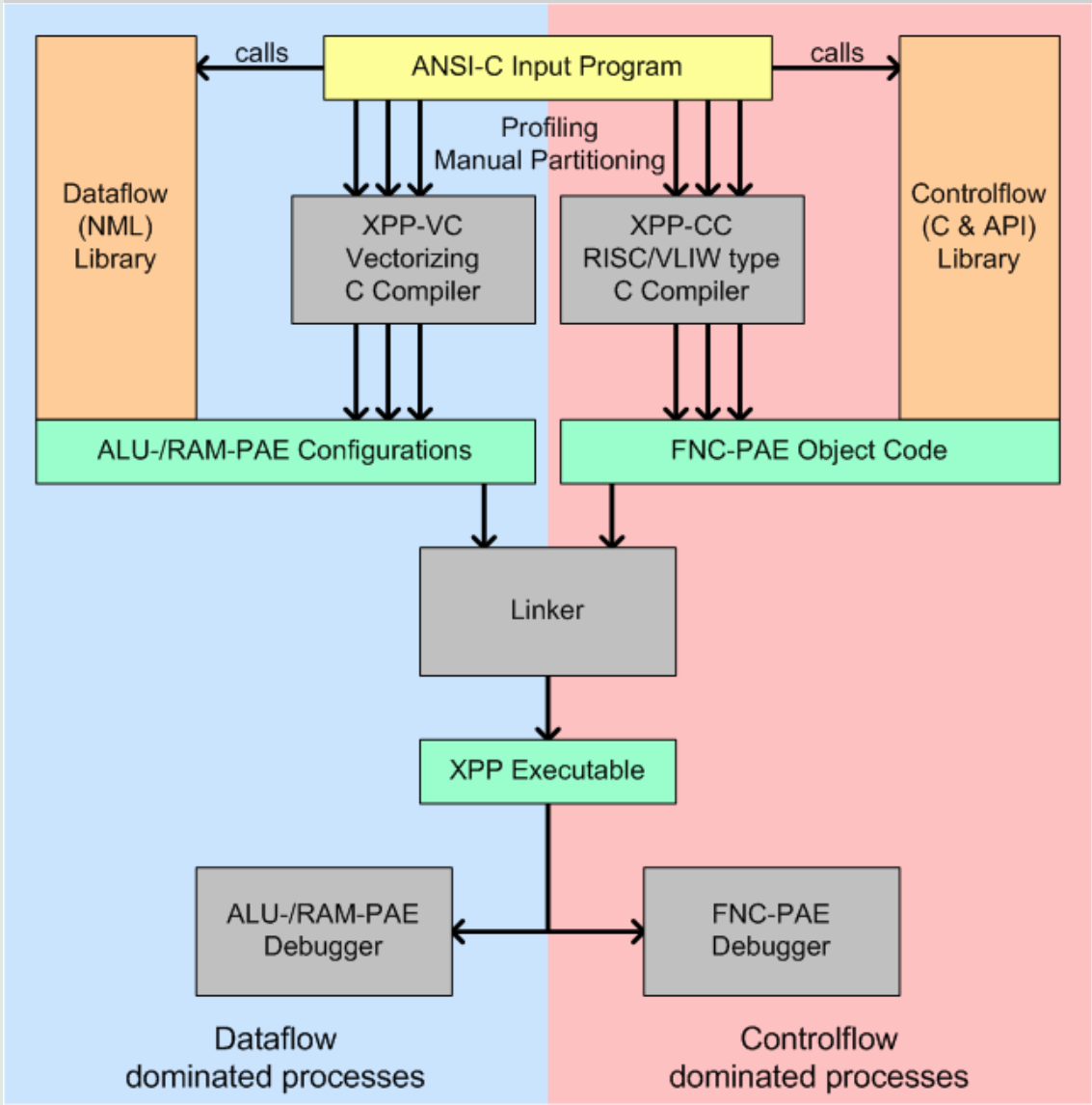
Mbits / sec	encode	decode
1 FNC	25	22

Context adaptive variable length decoder VLC (@100 MHz)

- Strictly sequential algorithm, for H.264
- 1 to 2 FNCs

Mbit/sec	
1 FNC	14
2 FNC	28

XPP-III Programming model



Add-on Tools:

- Profiler
- Partitioner: Annotation + automatic code generation

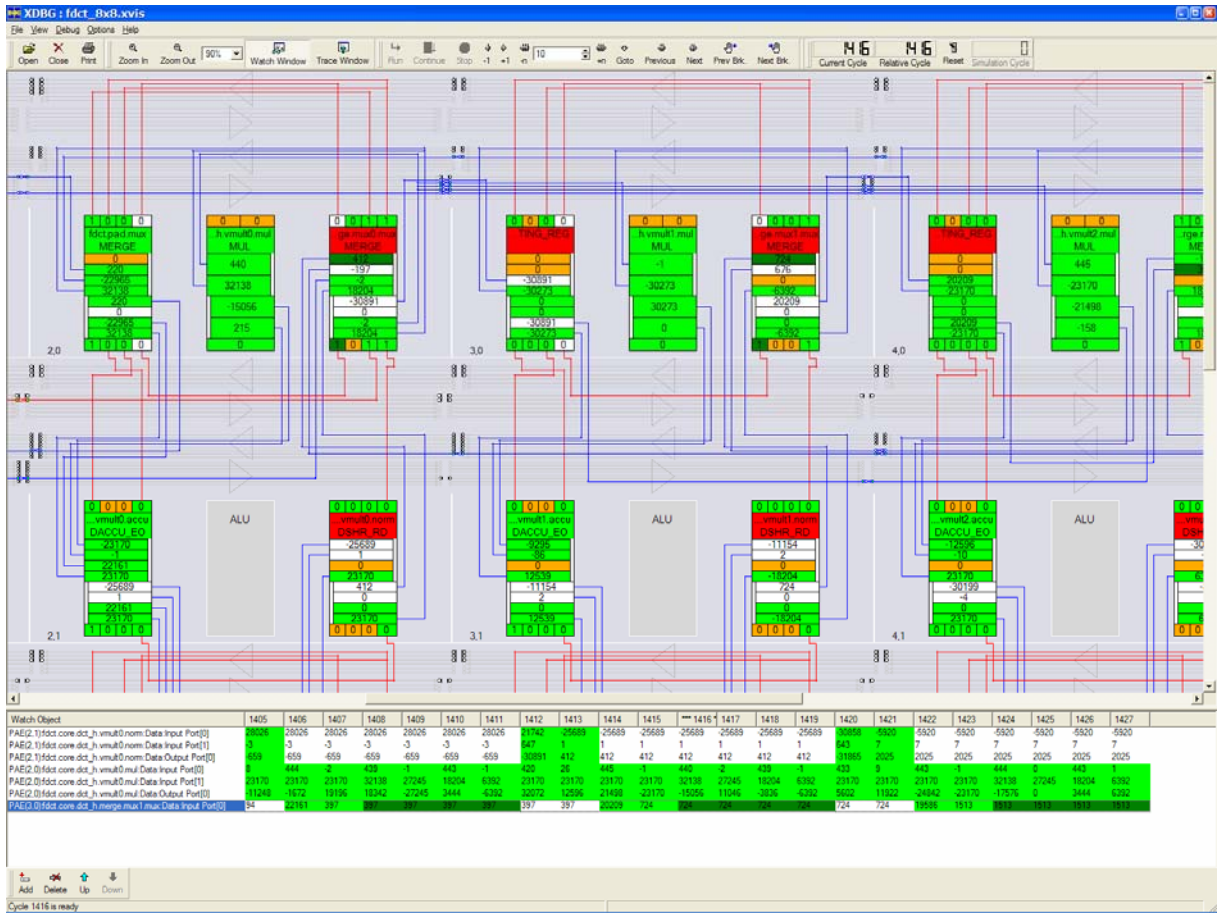
SystemC Models:

- Array*
- FNC-PAEs*
- Reference design*
- External Memory

* Cycle accurate model

Visual dataflow debugger

- Graphical view
- Same tool for Simulator and Hardware
- Hardware breakpoints, watch window

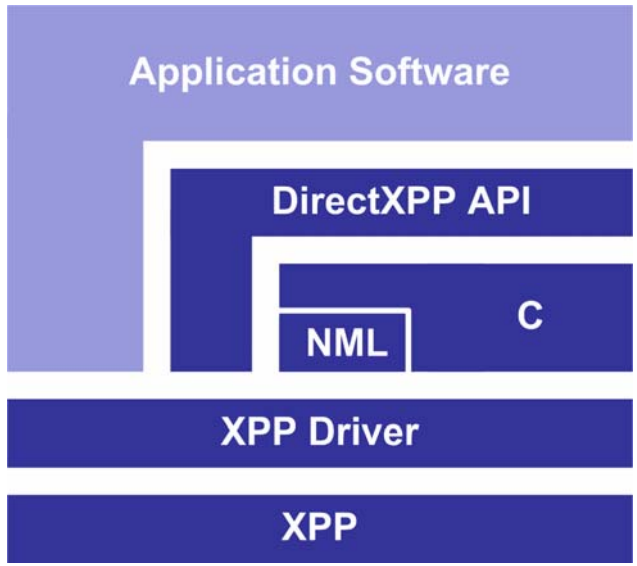


100% ANSI compatible C design entry

- Reuse of legacy code
- FNC-compiler supports also GNU C++.
- Easy code maintenance

High-Level Library and API

- High-level calls to complex application building blocks
  - E.g. SDR, Video, Error correction, Data En-/Decryption, etc
- Mid-level function calls to XPP core library
  - E.g. DCT, FFT, FIR, Image & Video Kernels …
- Low-level functions
  - Reconfiguration, Crossbars, DMA, Interrupt.



"DirectXPP"

- XPP abstraction layer for high-level function API
- Low-level API for direct access to XPP and reference design

XPP-3c Chip

XPP-3c Chip

Highlights

- XPP-III based fully programmable general purpose signal processor
- Very high internal bandwidth
- 6 Gbytes/sec External Memory bandwidth
- High speed interface 10 Gbps
- SDK, Simulator, C-compiler, Debugger

Applications

- Media Server Accelerator
- Multi-channel video transcoding
- HD Video codecs
- General purpose accelerators
- XPP-III reference platform,

Design targets

- Highest performance and bandwidth
- Maximum flexibility
- Testability and software debugging features
- Standard interfacing
- Power efficiency

XPP-3c Chip Features

Features

- 5x12 ALU PAEs, 24 RAM-PAEs, 350 MHz
- 12 FNC-PAEs, 200 MHz
- 2 X-RAMs. L2-caches: 64 KBytes each
- 4 4D Address Generators
- 3 external DDR2 RAMs
- High speed interface, 10 Gbit
- HD Video interface
- 4 x 16-bit XPP streams (async)
- JTAG software debugging
  - Array hardware breakpoints
- XPP-III Reference design with cycle accurate simulator

Physical design

- TSMC 90 G
- DDR2 controllers: Synopsys
- DFT and Bist
- Backend: Accent
- Samples: Q2 2008

More details:
NDA
required

Space Processor

XPP

Customization

- XPP-III IP building blocks can be parameterized and instantiated according to final system requirements. Streaming interfaces between all blocks handshaking protocols everywhere.
- Can range from stream processor to self contained DSP
- Some chip specific IP blocks may be required in addition:
  - System controller (PLLs, Reset, JTAG controller)
  - Memory controller(s)
  - Special interfaces (e.g. SpaceWire or Faster links) can easily be connected to the Sysmem Arbiter (high bandwidth) or the FNC-IO Bus (low bandwidth)

Additional building blocks

- Several wrappers have been implemented by PACT. They can be re-used or provide a basis for adapted designs.
  - AHB master and slave bridges (32-bit, async. clock domains) to Sysmem Arbiter
  - Asynchronous external XPP-stream interface (Stream-IO) with handshake (e.g. for array cascading, AD/DA …)
  - Wrappers to Synopsys DDR2 and SRAM / External Bus Controllers, EDAC option available.

XPP-III Based Space Processor

Physical Design

- IP is fully synthesizable: can be synthesized with Rad hard libraries (e.g. Atmel)
- Array has only a small amount of program memories reducing the probability for SEU which may harm the functionality.
- Internal interfaces: pipeline registers can be inserted wherever needed to simplify timing closure (no impact on software !) ECC memories possible.
- Hierarchical approach: Design hard macros for the array (e.g. ALU/RAM-PAE, FNC-PAEs → speeds-up the BE flow !

Low Power

- Array is implicitly low power due to minimum overhead. FNC-PAE clock can be scaled to application needs
- Currently unused elements (e.g. Address Generators, FNC-PAEs) can be set to idle mode.

Improved on-chip reliability strategies

- Array configuration can be re-written during operation
- FNC-PAE can re-calculate and compare array results by random examination or checkpoints.
- Cross supervision of processes by FCN-PAEs, background memory health check.

XPP-III Hardware IP Verification

Verifcation strategy

- FNC-PAE generates patterns and compares results → yes/no result simplifies verification during backend
- Test suite on FNC-PAE runs as cycle accurate SystemC model → much faster test execution compared to full RTL simulation
- Block-Level Verification test benches available
- Top level tests must be adapted to the final topology, can be written in C
- BFMs and block-level verification for chip specific interfaces are recommended
- All verification routines can be used for functional chip tests and health check during operation.

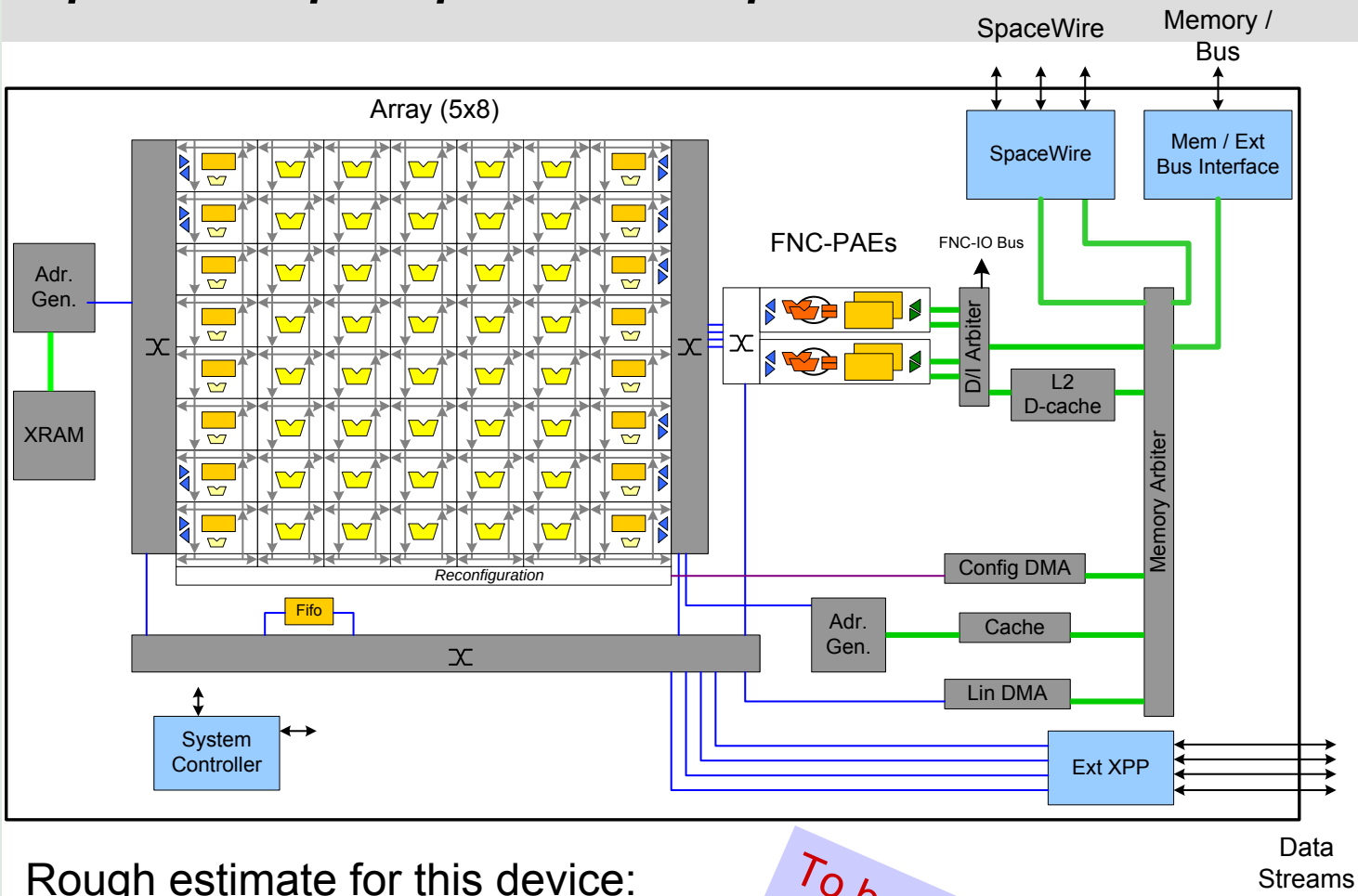
Limited Hardware integration effort due to

- Silicon proven IP
- Plug & play building blocks
- Hierarchical backend
- Self-contained test benches
- SystemC models

IP Availability

- RTL of XPP Array, FNC-PAEs: now
- RTL of all Reference design modules: Nov. 07
- C-Compiler: now
- Complete cycle acc. Software tools for the XPP-3c (XPP-3 SDK): Q1 2008

A possible space processor implementation



Rough estimate for this device:

- Array: 3 MGates, 131kbit RAM
- FNC-PAEs: 310 kGates, 700 kbit RAM
- Ref design: 900 kGates, 400 kbit RAM
- Other blocks estimate: 300 kGates

To be adapted to final requirements !

Summary

XPP combines

- Coarse grained dataflow array
- + Sequential VLIW DSP cores
- + Communication framework
- + Complete tools-suite (C, C++, native input, simulator, tools)

Outlook

- Array: Floating Point ALUs-PAEs
  - Replace fixed point ALUs by pipelined FP cores.
  - Array area: + 20.. 30%.
  - No substantial change in architecture, software and tools
  - Format conversion in links to fixed-point FNC-PAEs
  - XPP40FP: 8 GFlops @100 MHz
- FP Availability
  - IP is defined and evaluated
  - Final implementation on demand

Thank you !

XPP in space processors

Fully programmable

Very high parallel performance

Wide application space

Easy IP integration

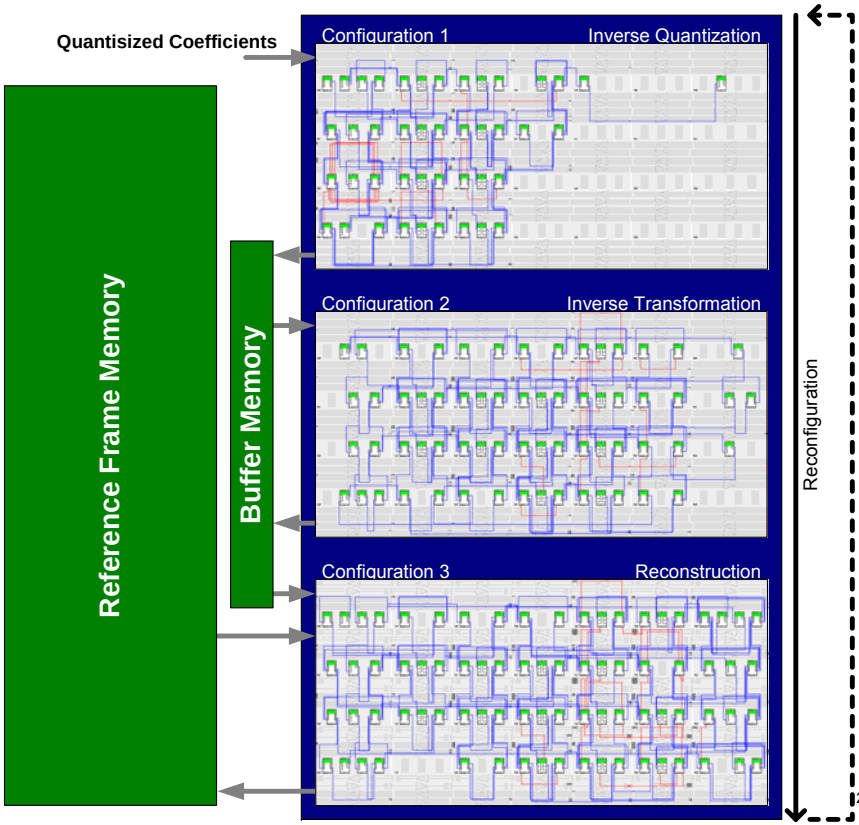
Scalable

Complete tools

www.pactxpp.com

Backup

XPP Execution and Reconfiguration Scheme



- Typically 40−80 MacroBlocks are processed per configuration
- Intermediate data is exchanged via Buffer Memory between the configurations
- After sequence of configurations is completed, processing re-starts at configuration 1 with next set of MacroBlocks

Reconfiguration Overhead

Theoretical minimum

At a buffer size of 40-80 MacroBlocks reconfiguration overhead becomes negligible