Application Specific Processors in Industry SoC Designs

MPSoC '04

Steffen Buch

Senior Director
Advanced Systems & Circuits

Infineon technologies

Never stop thinking.

Infineon

Agenda

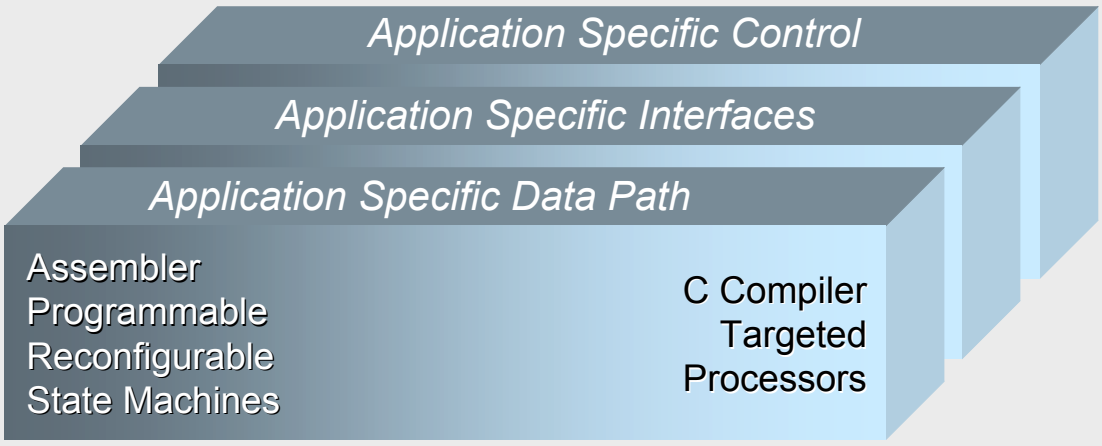
- Motivation
- Example 1: Filter Development Platform – ASMD
- Example 2: Network Processor Core – PP32
- Conclusions

Never stop thinking

Agenda

- **Motivation**
- Example 1: Filter Development Platform – ASMD
- Example 2: Network Processor Core – PP32
- Conclusions

What Means Application Specific Processor ?

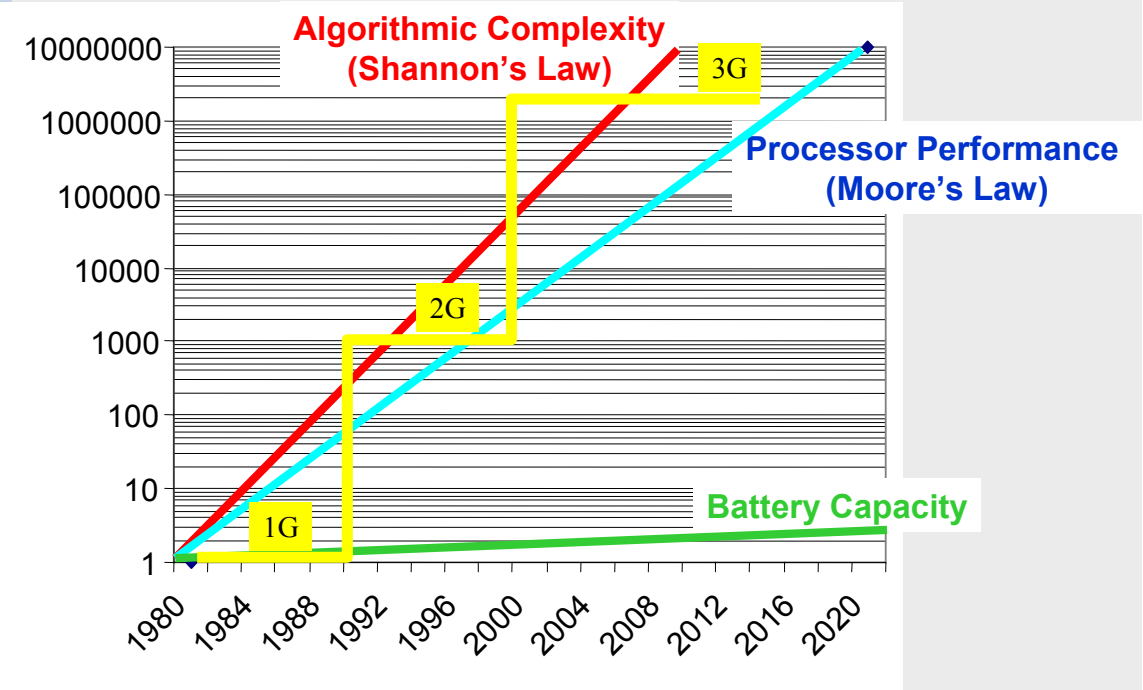


- Application Specific Processors cover a broad range of different flavors
- Different point of views from the hardware and the software world open up new possibilities

4 Drivers for Application Specific Processors

- ✓ Performance Requirements
  - throughput
  - power efficiency
  - code & data density (memory requirements)

Performance Driver: Shannon's Law



MIPS Requirements for Wireless Technologies

• DECT	10 MIPS
• Bluetooth	20 MIPS
• GSM	100 MIPS
• GPRS	350 MIPS
• EDGE	1200 MIPS
• UMTS	5800 MIPS

Next Challenge: MIMO

⇒ **Advances in air interface technology (higher data rates, higher mobility, more flexibility ...) significantly increases processing requirements**

Source: Dr. J. Hausner, VP Concept Engineering, Secure Mobile Solutions, Infineon

Signal Processing for an UMTS Receiver (Air Interface)

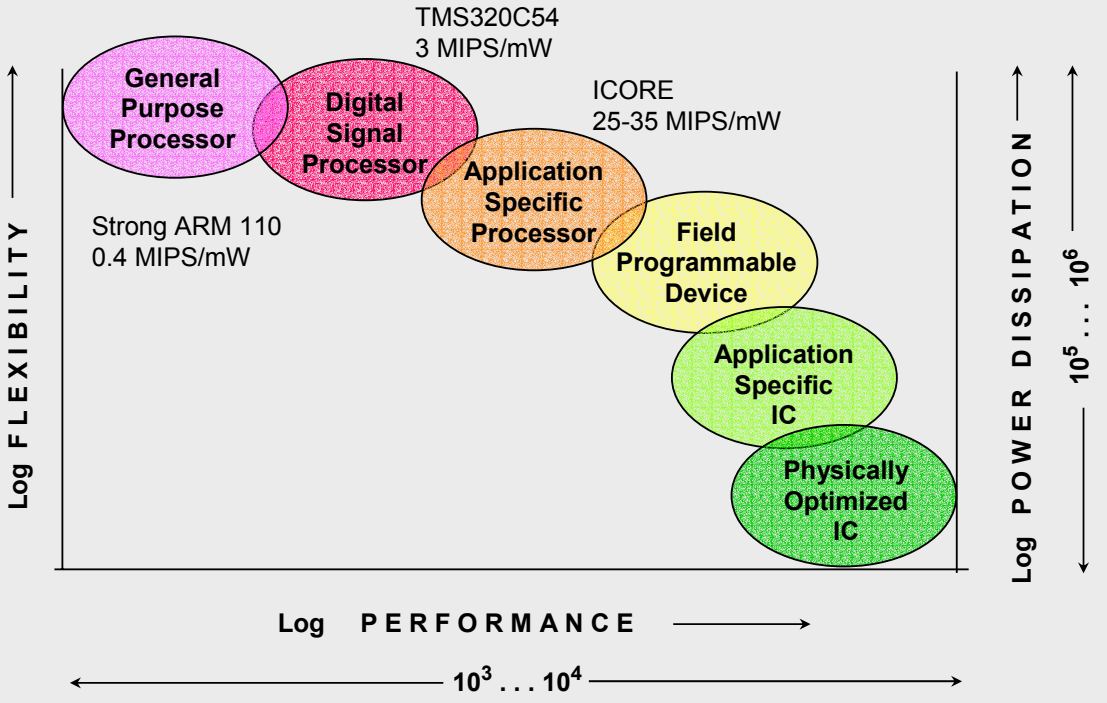
Example of processing requirements @ 384 kbps :

Digital Filtering (RRC, channelization)	~3600 MIPS
Searcher (frame, slot, delay path estimation)	~1500 MIPS
RAKE receiver	~650 MIPS
Maximum-ratio combining (MRC)	~24 MIPS
Channel estimation	~12 MIPS
AGC, AFC	~10 MIPS
Deinterleaving, rate matching	~14 MIPS
Turbo-decoding	~52 MIPS
Total	**~5860 MIPS**

⇨ **UMTS requires intensive layer 1 processing compared to e.g. GSM**

Source: Dr. J. Hausner, VP Concept Engineering, Secure Mobile Solutions, Infineon

Flexibility - Performance - Power Trade-Off

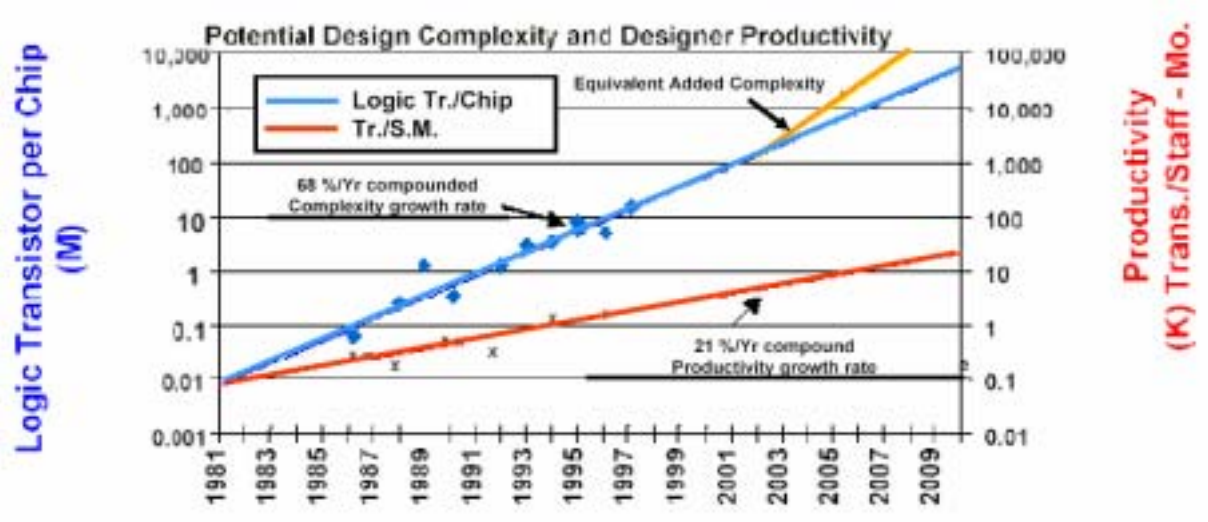


Source: T. Noll, RWTH Aachen

4 Drivers for Application Specific Processors

- ✓ Performance Requirements
  - throughput
  - power efficiency
  - code & data density (memory requirements)
- ✓ Design Efficiency
  - platform approach: easy adaptability to different applications in same domain
  - predefined control & pipeline structure simplifies design

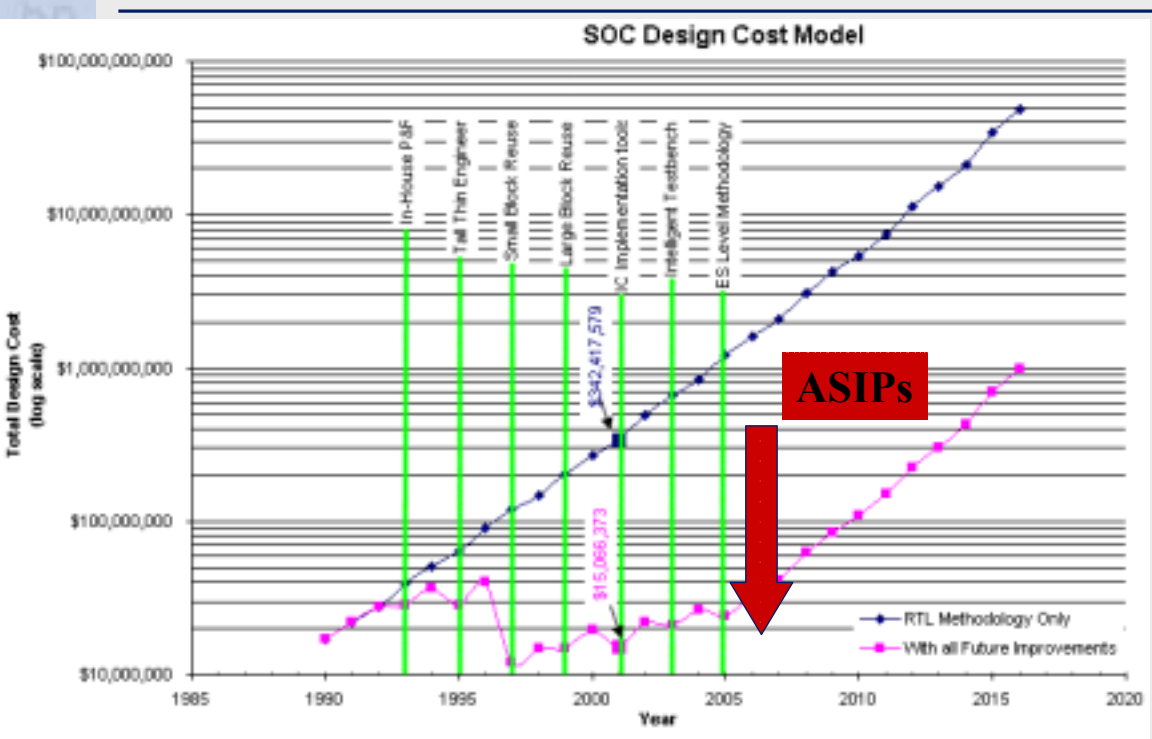
Driver for Better Design Efficiency: Moore's Law



Source: ITRS

Does anybody not know this slide?

Leading Edge SOC Design Cost



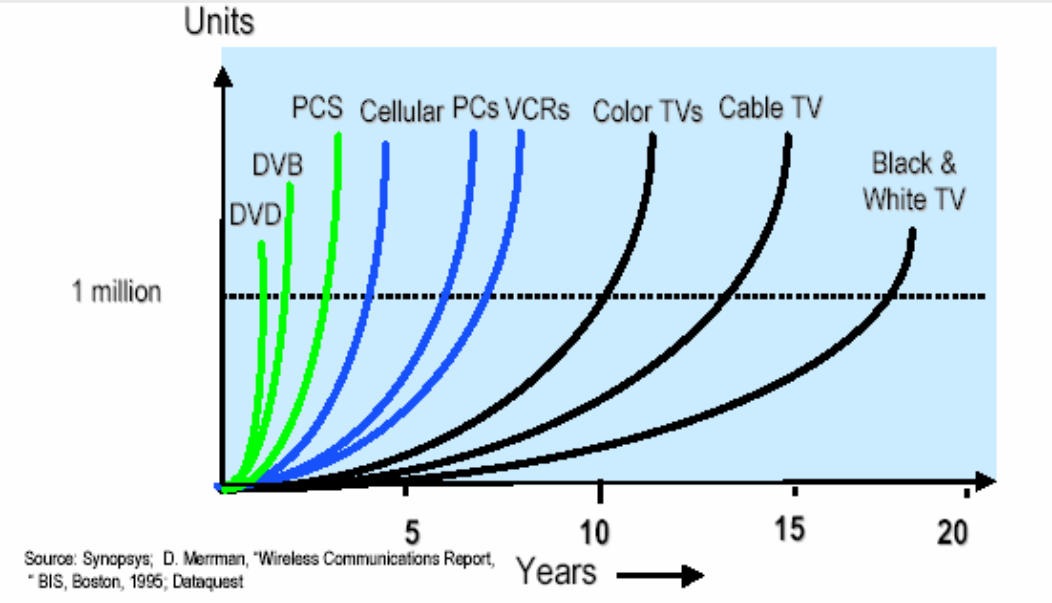
Source: International SEMATECH

Note: Cost called out are for 8M gate PDA in 2001

4 Drivers for Application Specific Processors

- ✓ Performance Requirements
  - throughput
  - power efficiency
  - code & data density (memory requirements)
- ✓ Design Efficiency
  - platform approach: easy adaptability to different applications in same domain
  - predefined control & pipeline structure simplifies design
- ✓ Product Programmability
  - late/in-field changes
  - product derivatives
  - bug fixing

Drivers for Programmability in SOC



- Fast evolving standards
- Unstable "standards"
- Concurrent engineering with customers
- Platform derivatives from one base chip (reduce NRE)

4 Drivers for Application Specific Processors

- ✓ Performance Requirements
  - throughput
  - power efficiency
  - code & data density (memory requirements)
- ✓ Design Efficiency
  - platform approach: easy adaptability to different applications in same domain
  - predefined control & pipeline structure simplifies design
- ✓ Product Programmability
  - late/in-field changes
  - product derivatives
  - bug fixing
- ✓ IP Cost
  - ASIP design allows quick and cheap IP development

IP Cost Can Eat-Up Your Gross Margin

- Extensive IP licensing leads to high up-front & royalty payments
- Gross margins are lowered
- IDMs or design houses can not be successful by only integrating licensed IP

Example:

!!! France Telecom daughter company demands about 1$!!!
!!! per chip from 3G chip makers for Turbo Coding !!!

Con's for Application Specific Processors

- ✘ Design Effort (Tools + Core)
  - Requires new methodologies

Requirements for ASIP Design Methodology

- Easy profiling of applications
- Short processor design iteration loops
- Efficient development of tool environment and hardware
- One source for hardware and software views
- Consistent verification environment
- Support of reuse concepts
- Support for documentation

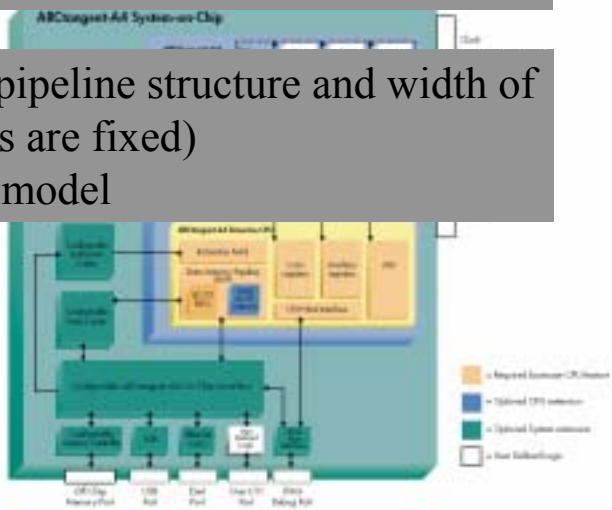
Two Approaches to ASIP Design

- Extendible pre-defined processor cores
  - customize basic instruction set with user defined extensions
  - examples: Tensilica, ARC
  - ARM and MIPS also offer extensions

- Language based processor design
  - describe processor architecture and instruction set architecture in one language
  - generate tool chain and HDL code from description
  - C-like syntax with extensions (nML)
  - examples: Target Compilers (Chess, Checkers)
    CoWare (LISATek)

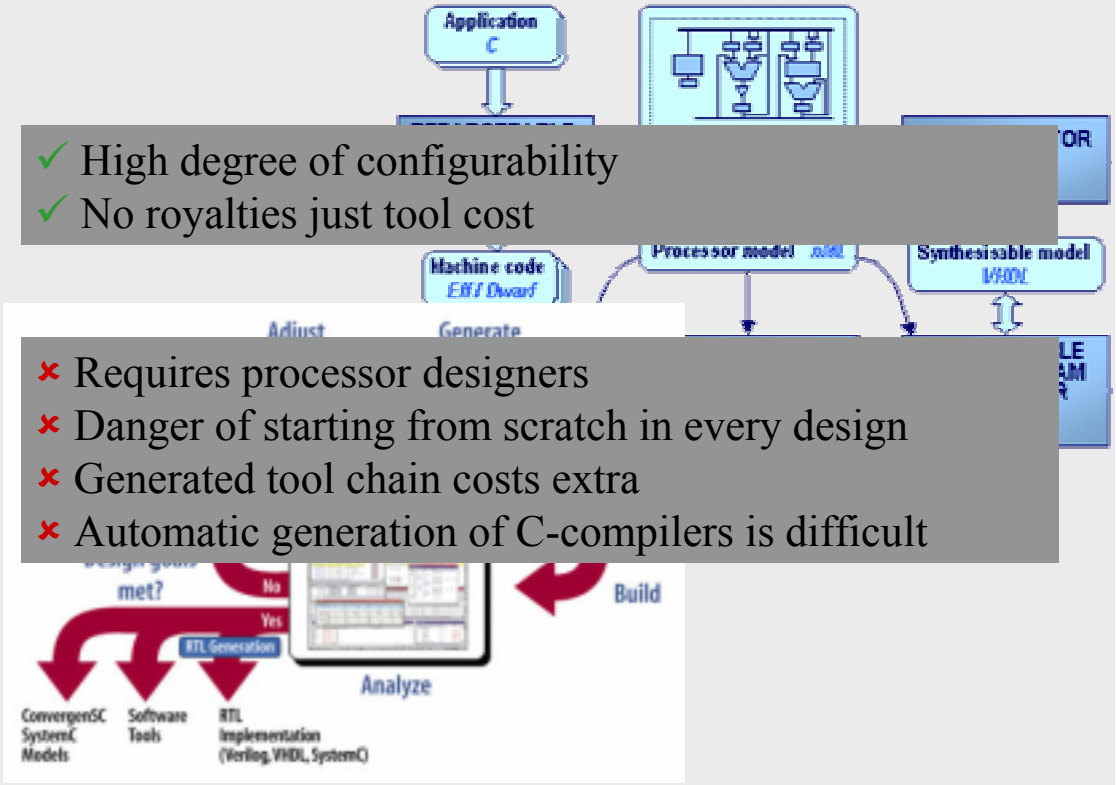
Extendible Processors

✓ Only limited processor design know how required
✓ Pre-defined modules speed-up design
✓ Better C-compilers



✗ Limited flexibility (e.g. pipeline structure and width of registers and instructions are fixed)
✗ Royalty based licensing model

Language Based Processor Design



- ✓ High degree of configurability
- ✓ No royalties just tool cost

- ✗ Requires processor designers
- ✗ Danger of starting from scratch in every design
- ✗ Generated tool chain costs extra
- ✗ Automatic generation of C-compilers is difficult

Con's for Application Specific Processors

- ✗ Design Effort (Tools + Core)
  - Requires new methodologies
- ✗ Processor Verification
  - Requires new methodologies

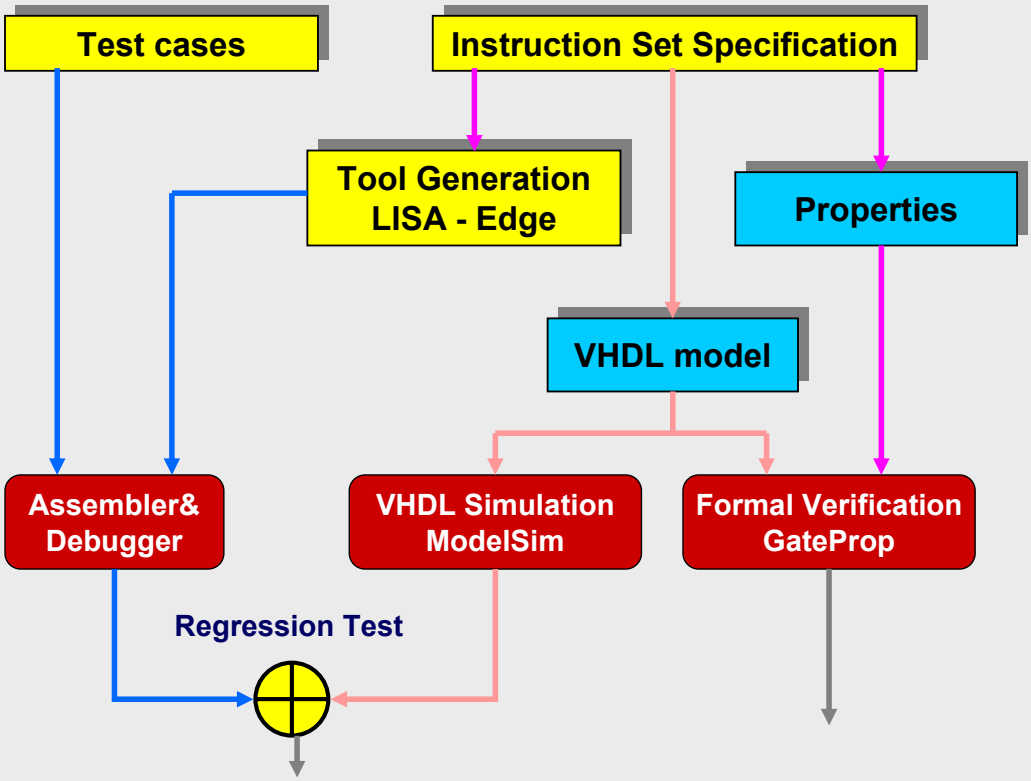
Processor Verification

- ASIP verification is as difficult as general purpose processor verification
- Exception: application is verified together with processor

Two complimentary solutions:

- Constrained random pattern testing (configurable Genesys)
  - IBM expert is here: Roy Emek
- Formal verification in processor design
  - Application of property checking

IFX ASIP Verification Flow



Con's for Application Specific Processors

- ✘ Design Effort (Tools + Core)
  - Requires new methodologies
- ✘ Processor Verification
  - Requires new methodologies
- ✘ Programming Models for Multi-Processor Systems
  - No general solution existing yet
  - Further research required

Programming Models for Multi-Processor Systems

- Partitioning and verification of complete system in multi-processor architectures are challenging
- Tendency of **hardware designers**:
  Use multiple (different) processors for efficiency reasons
- Tendency of **software designers**:
  Use single core solutions because that simplifies software/firmware development

☛ Generalized transparent programming of embedded MPSoCs would be the perfect solution BUT optimization problem is not solvable

☛ Constrain yourself to typical application scenarios

☛ Two typical scenarios
- Two core designs
- One master core + flexible coprocessors

Two Typical MPSoC Scenarios

1. Two core solutions (example Controller + DSP)
   - Program from **one source code**
   - Manual partitioning (annotation in source code)
   - Inter-processor communication generated by compiler automatically
   - Compiler friendly IPC

2. One master core and multiple programmable slave processors
   - Program from **one source code**
   - Either manual or automatic partitioning
   - Communication automatically generated

??? Are there any solutions around ???

Agenda

- Motivation
- Example 1: Filter Development Platform – ASMD
- Example 2: Network Processor Core – PP32
- Conclusion

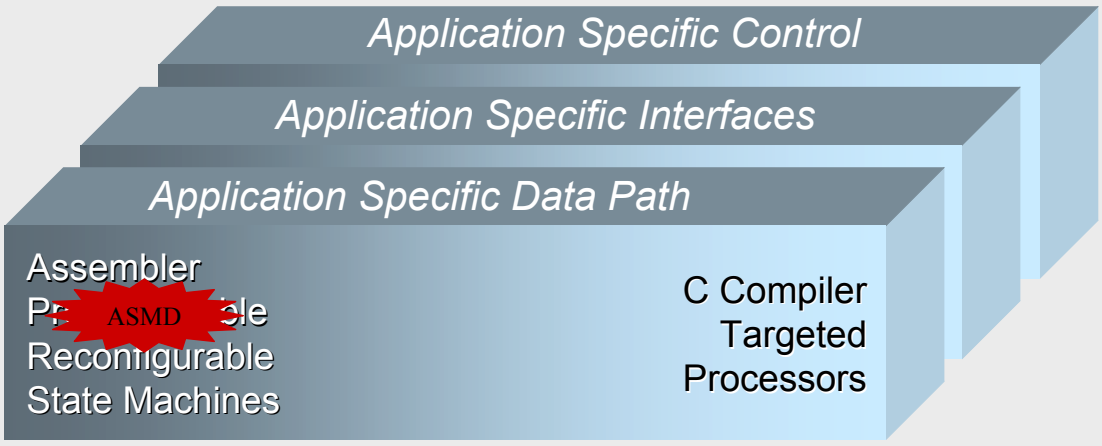
Design Goal:
Expert System for Digital Filter Design

ASIP core + HW accelerators + glue logic for digital part in mixed-signal front-ends (e.g. audio codecs, transceivers)

Requirements:

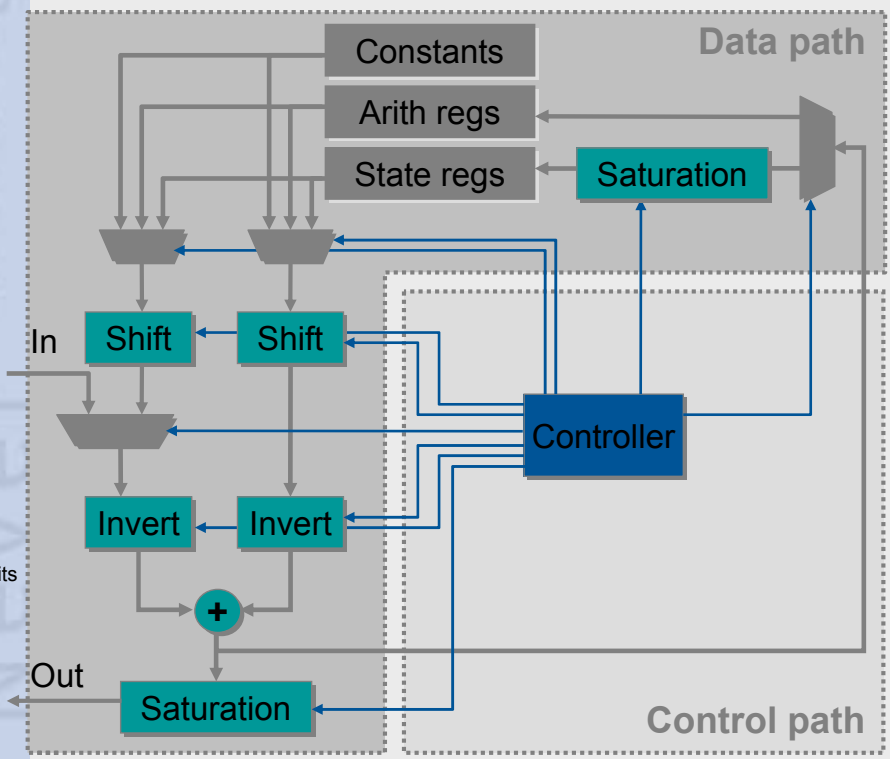
- High data rates with limited cycle budget and high power sensitivity
- Typical cycle budget is 50 to 500 cycles per sample
- Typical clock speed is about 100 MHz
- Typical tasks are multi-rate interpolation/decimation, filtering, coding and mixing
- Short design time, quick adaptability

Application Specific Multi-Rate DSP - ASMD



- Major driver for ASIP development
  - design efficiency
- May be not even a processor but a flexible state machine

ASMD Architecture



- Shift-and-Add DSP
- Optimized for data flow operations
- Conditional operations
- Highly parameterised
- Assembler creates control sequence for data path
- Fully synthesizable

Assembler Syntax and Instruction Set

example_lisa.asm

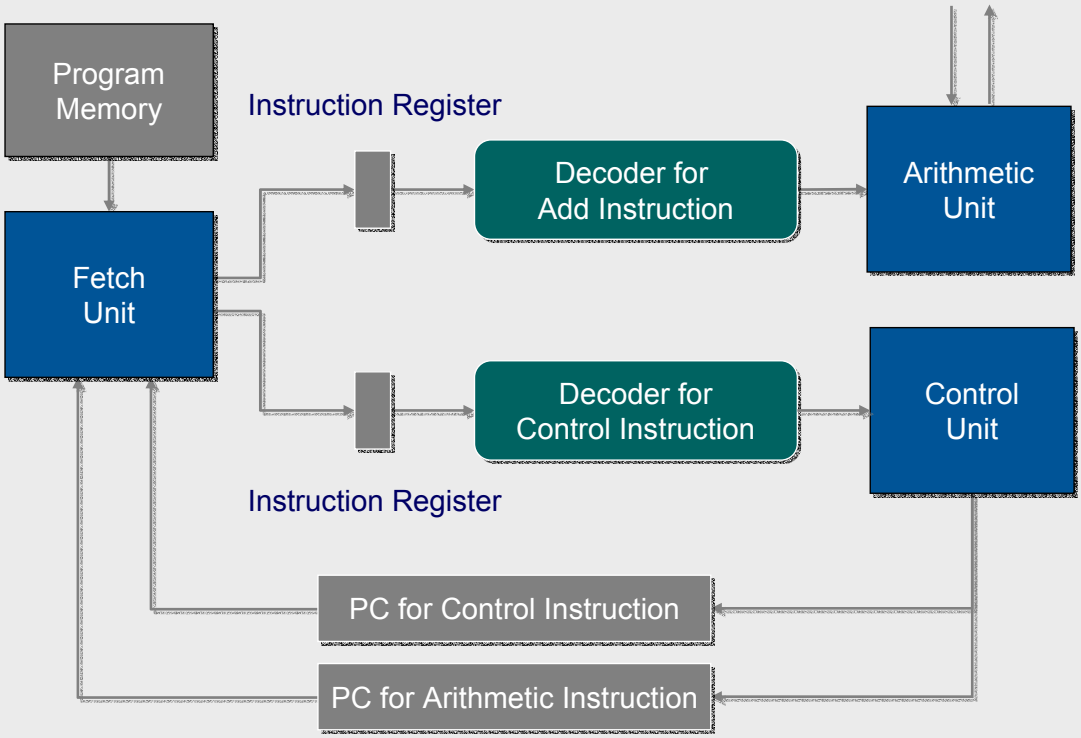
```
        .text

Program_test:
        !Subprog1 from label DA16b to label DA16end slic      g
        !Subprog1 from label DA16a to label DA16a_e s      end_fla

Subprog1:
DA16a:
        state12, state1,  shift0, shift0,      ), arith1
        arith1,  arith1,  shift1, sh          +(+), arith2
        arith1,  arith2,  shift0,             -(-), arith2
DA16a_e:
        arith1,  arith2,          ft0, +(+), -(-), state13, out1

DA16b:
        in1,              hift0, shift0, +(+), +(+), state12
        state12,          , shift0, shift2, +(+), -(-), state12
        state12, s ate4,  shift0, shift0, +(+), +(+), arith1
DA16end:
        arith1,  arith2,  shift0, shift0, +(+), -(-), state13, out1

        .end
```

1st cycle
6th cycle
2nd cycle
3rd cycle
4th cycle
5th cycle

Poor utilization of data path

Parallel execution of two instructions

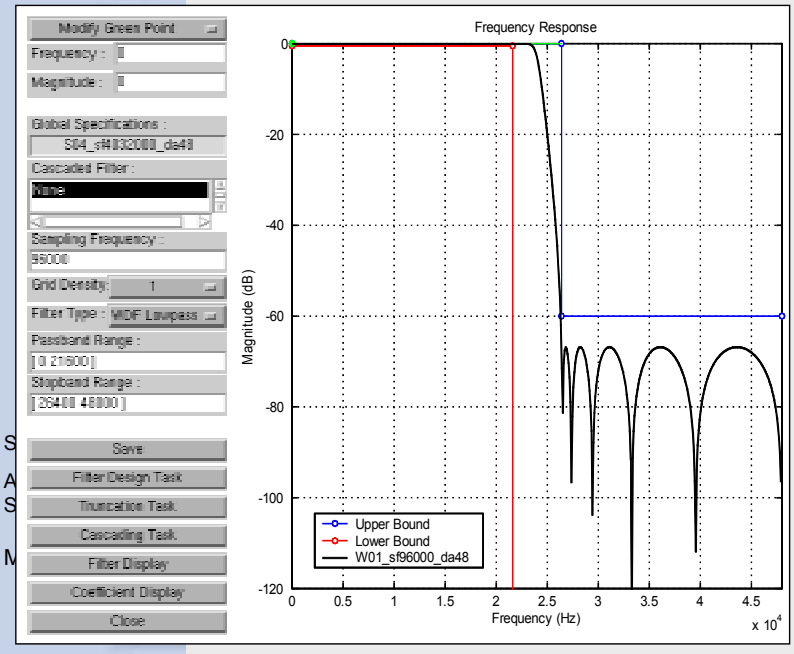
Parallel Control and Data Flow



Steffen Buch
Advanced Systems & Circuits

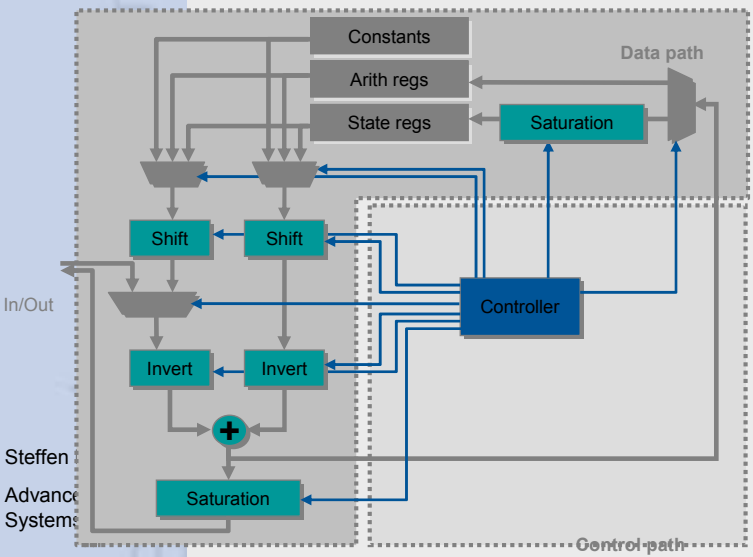
MPSoC'04

Expert System for digital filter front-ends



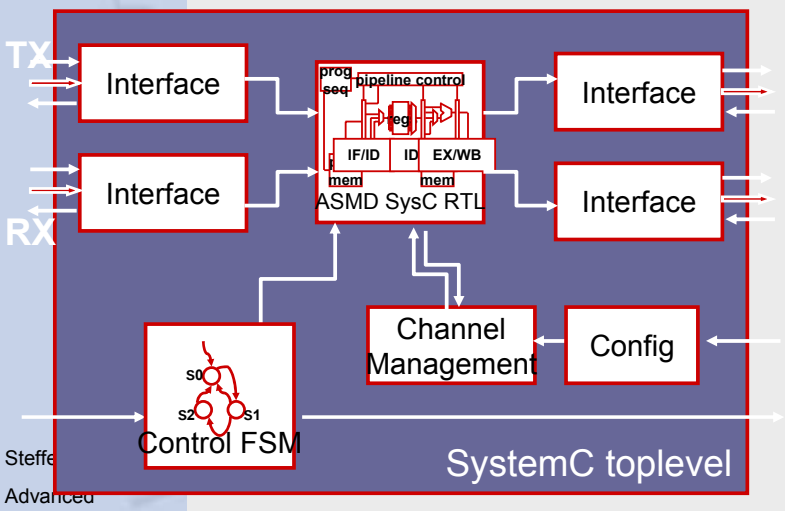
1. Algorithm development in Matlab environment

Expert System for digital filter front-ends



1. Algorithm development in Matlab environment
2. Configure and program ASMD using software macros

Expert System for digital filter front-ends



1. Algorithm development in Matlab environment
2. Configure and program ASMD using software macros
3. Design complete module including HW accelerators and glue logic using pre-designed libraries

- Average design time could be reduced from 4 to 1 month
- Flow applied in several projects:
  GSM, UMTS, Bluetooth, VoIP, automotive multi-media

Productive Target Application for Code Generation

CVSD Speech Codec for Bluetooth

Requirements:

- Three channels for receive and transmit each with undo functionality
- 26 MHz clock frequency

➢ Design based on ASMD core modeled with LISA

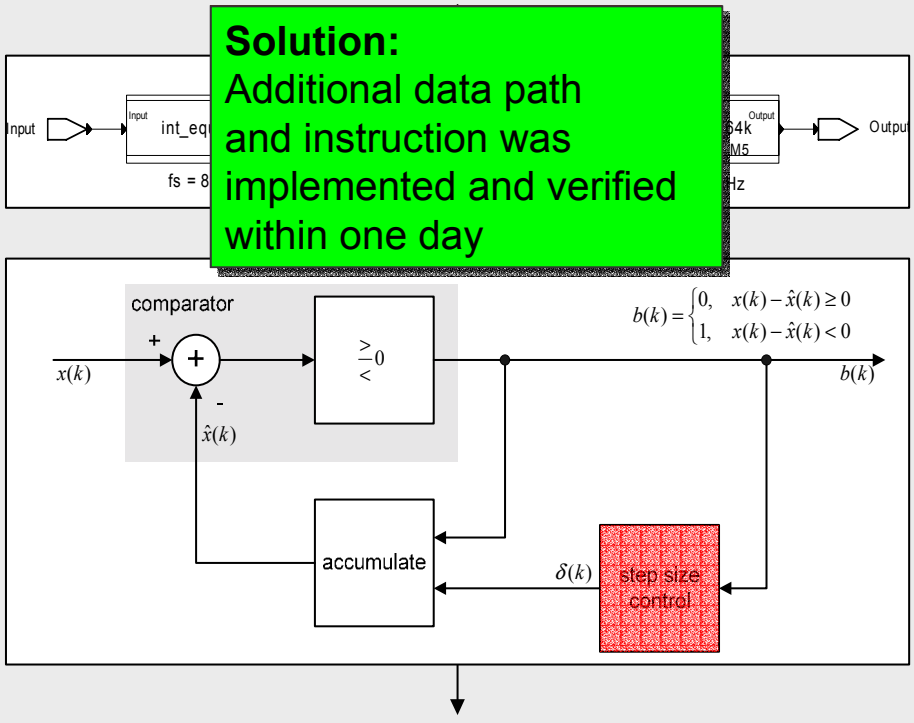
➢ Research cooperation with RWTH Aachen on SystemC RTL generation

CVSD Transmit Path

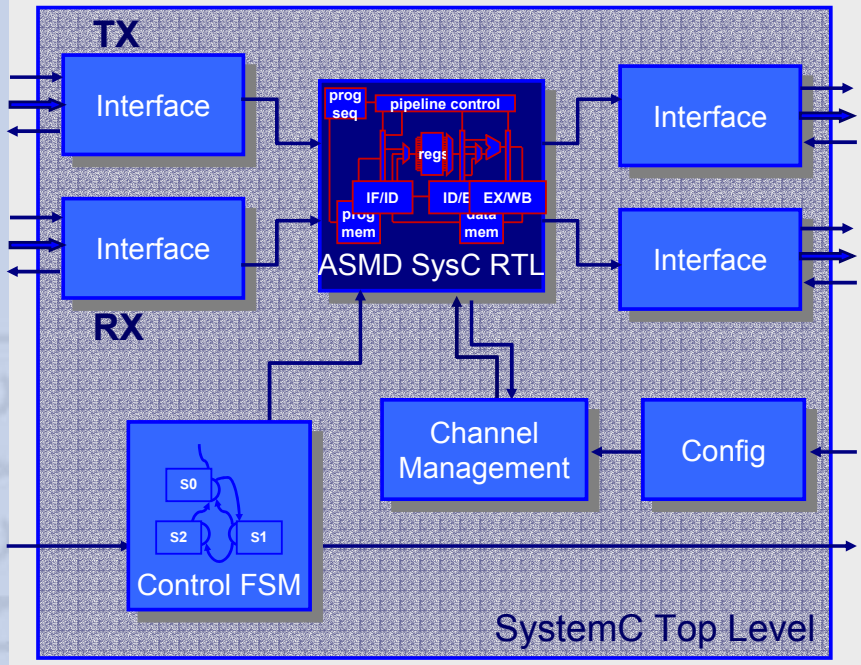
Inter-polation Filters

CVSD Encoder

Solution:
Additional data path and instruction was implemented and verified within one day

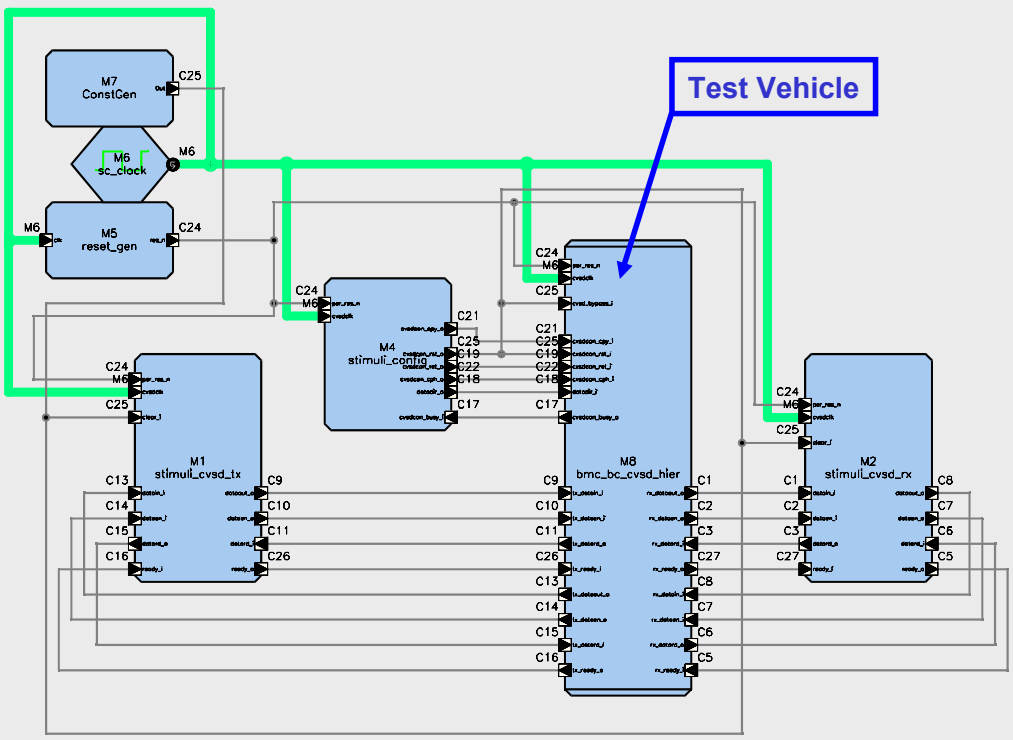


Hardware Partitioning / Design Environment

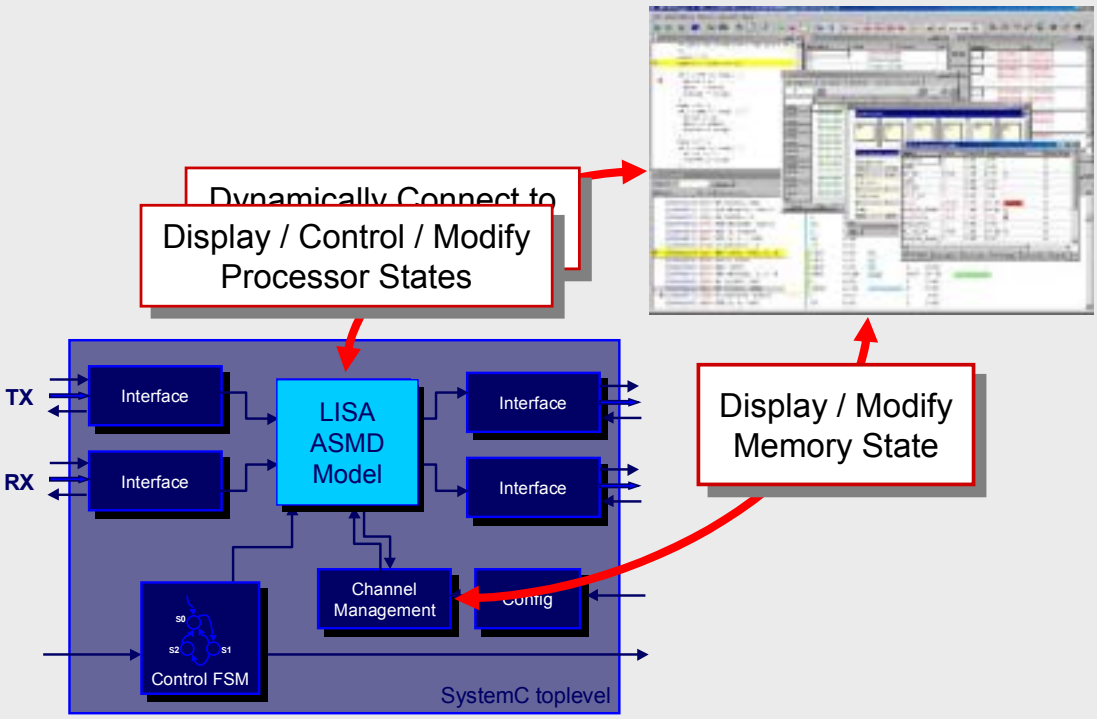


- SystemC 1.0.2 testbench
- Design of glue logic in SystemC RTL
- Cross compilation to VHDL
- Synthesis with Design Compiler

SystemC Verification Testbench



Hardware / Software Co-Verification



Agenda

- Motivation
- Example 1: Filter Development Platform – ASMD
- **Example 2: Network Processor Core – PP32**
- Conclusions

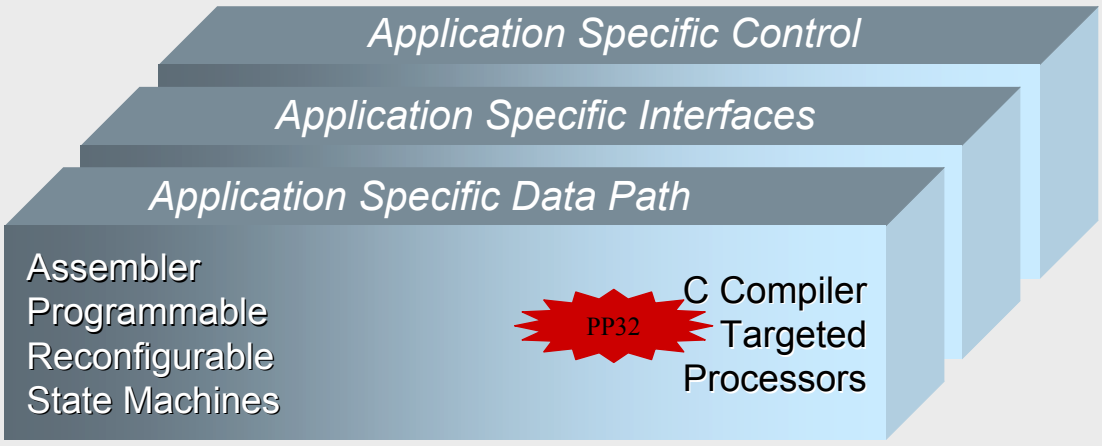
Design Goal:
Efficient Core for Framing Applications

Layer 1 and 2 packet processing in communication systems that require medium to high data throughput and flexibility.

Requirements:

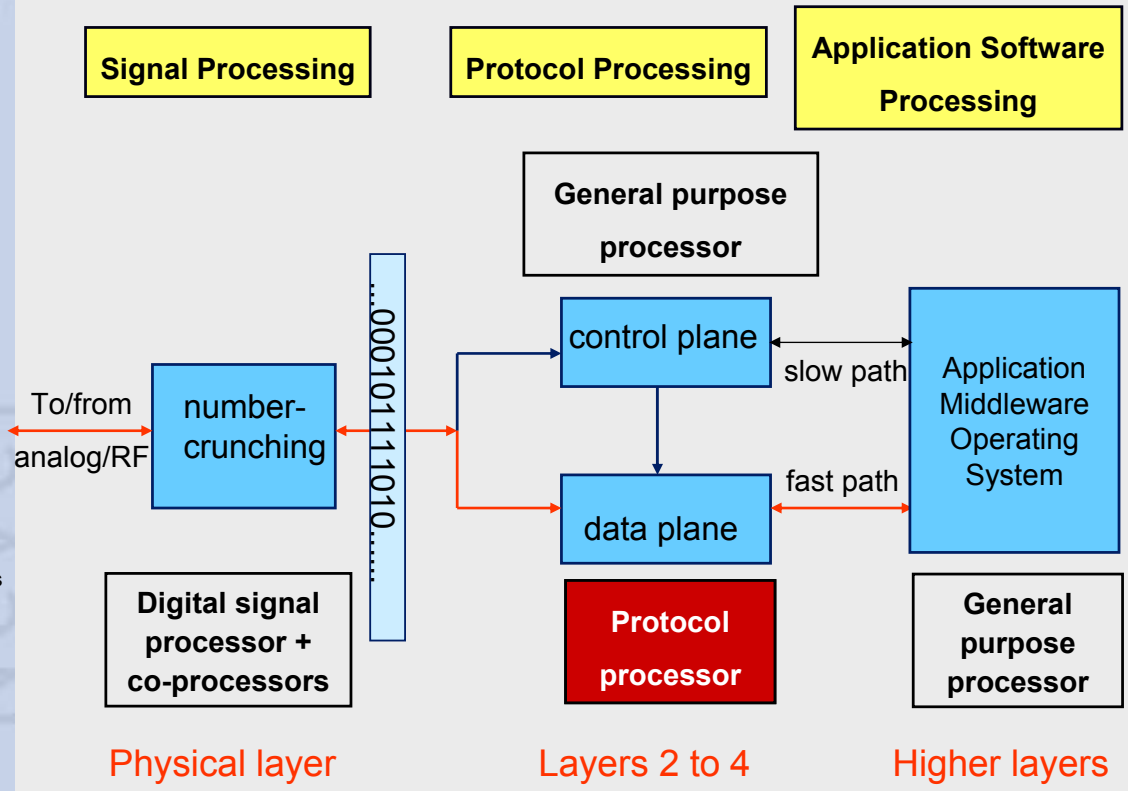
- Adaptability to different and/or changing protocols
- Optimized for hardware – software interaction
- Strong support of data interfaces
- Special features for protocol processing
- Predictability of execution time

32 Bit Protocol Processor – PP32



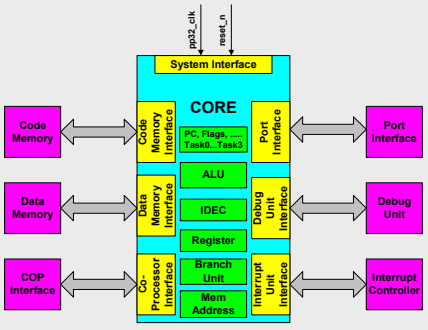
- Major driver for ASIP development
  - performance requirements
  - product programmability
  - IP cost

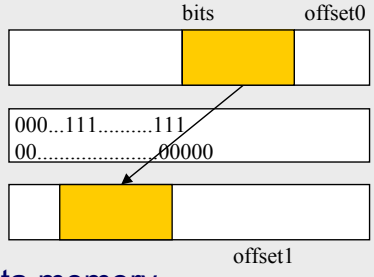
Principal Receiver Partitioning



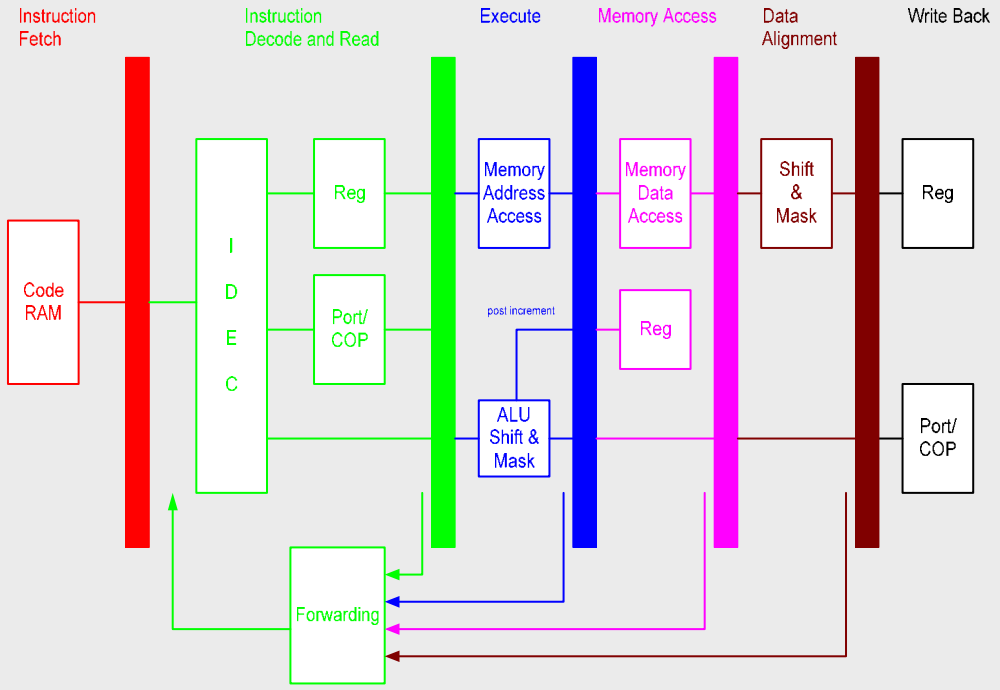
PP32 Architecture Overview

- 32-bit RISC architecture with 6-stage pipeline
- Harvard architecture with separate code/data buses
- Hardware supported multithreading
  - Up to 4 threads can be implemented
  - Each thread has its own register/flag space
  - 16 x 32 bit registers per thread
  - Low-overhead thread switch
- Extensible and open architecture
  - Up to 16 register mapped ports
  - Behaviour of each port is user-defined
  - Support of scratch memory and global data memory
- Extensive support for bit field manipulation





PP32 Pipeline Architecture



PP32 Instruction Set: Overview

- Data transfer operations
  - reg-reg, reg-port, port-reg, port-port
  - reg-mem, mem-reg, port-mem, mem-port
  - Bit-field transfer
- Arithmetic and logical operations
  - Operand either register or immediate
- Branch and thread control
  - Thread switch with and without priority
- Conditional execution of almost all instructions

☛ **Lean ISA: only 40 instructions**

PP32 Instruction Set Performance Comparison PP32 vs. MIPS M4K

The worst case example of the 10 test scenarios selected from L2 switching:

```
PROCESSING()
{
read packet start addresses
build the CAME keyword
read packet end addresses
launch the CAME look up
wait for CAME results
read the results
find what for a case we have
edit the packet according to this case :
strip VLAN
edit and send the ATM header
add LLC header
edit AAL5 trailer
send the rest payload
write cmd to the FIFO control
}
```

	Code Size	Performance
PP32	38x32 bits	38 cycles
MIPS M4K	56x32 bits	57 cycles

47% larger

50% slower

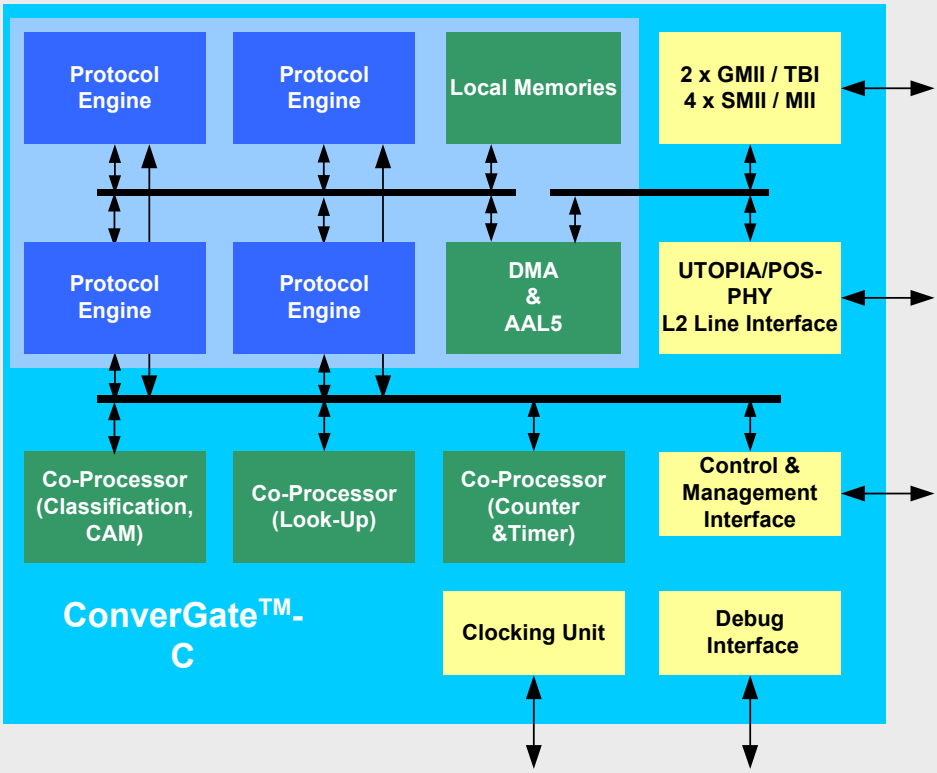
- Higher clock speed
- Smaller core size
- No license fees & royalties

PP32 Implementation

- Fully developed in the Infineon design flow (inway)
- Fully synthesizable (standard-cells) design
- Fully synchronous interfaces

Technology	***0.13 µm***
Design Package	c11n v2.1.2
Operation Condition	worst case (1.35 V, 125 °C)
Library	cstarlib_reg_1v5
Synthesis result	**Frequency**: 303 MHz in Worst Case (c11n RVt 1.35V 125C) **Area**: 0.37 mm^2
Deliverables	VHDL model, Synopsys DC synthesis scripts
Tools	Simulator, Debugger, Assembler, Linker, C-Compiler

ConverGate™-C V1.1 Block Diagram



Steffen Buch
Advanced Systems & Circuits

MPSoC'04

Agenda

- Motivation
- Example 1: Filter Development Platform – ASMD
- Example 2: Network Processor Core – PP32
- Conclusions

Steffen Buch
Advanced Systems & Circuits

MPSoC'04

Conclusions

ASIPs are the next revolution in SoC design !

Further improvements of ASIP methodologies and business models

Research and solutions for simple embedded MPSoC programming

Steffen Buch
Advanced Systems & Circuits