

ASIPs: Programmable Accelerators for Multicore SoCs

A High-Throughput JPEG Encoder Case Study













Steve Cox, Gert Goossens, Erik Brockmeyer
Target Compiler Technologies
steve.cox@retarget.com

SoC Conference, Newport Beach, CA
November 4, 2009

Agenda









- ▲ **ASIPs and Target – introductions**
- ▲ **High-throughput JPEG encoder design**
 - Specification
 - Multi-core partitioning
 - DCT ASIP
 - VLC ASIP
 - Evaluation
- ▲ **Conclusions**

Who are we?

    Audio	    Video & Multimedia	     Wireless
--	---	--

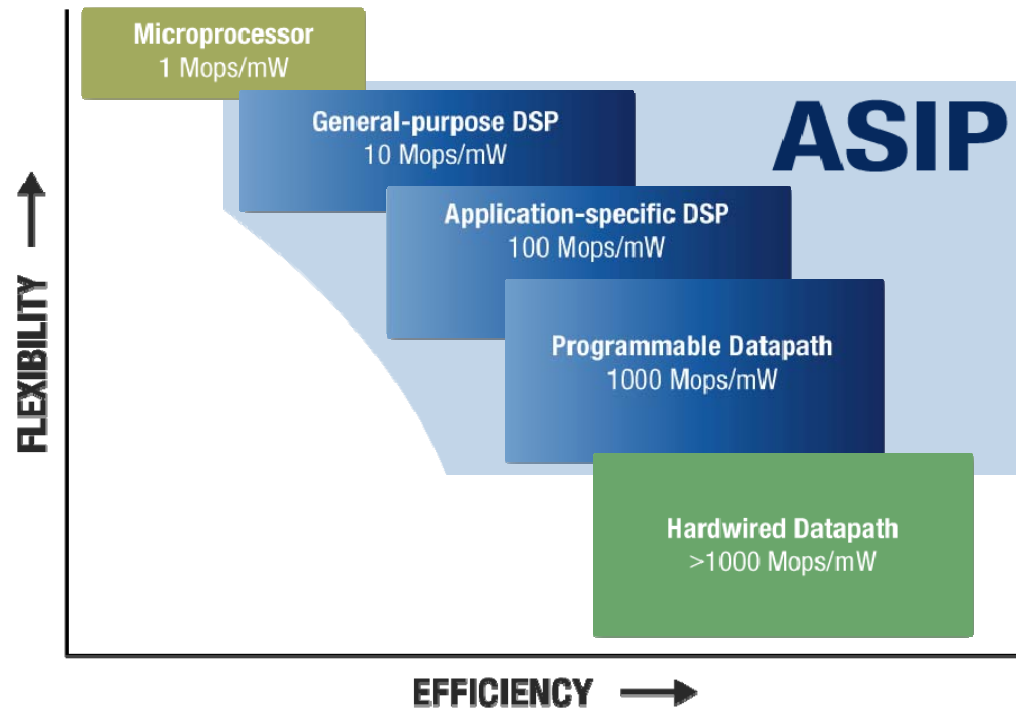
- ▲ Leading provider of EDA tools for application-specific processors (ASIPs)
- ▲ No Royalties!

	Automotive	IP Designer/ IP Programmer	Wireline	
---	-------------------	---------------------------------------	-----------------	---

Medical    	HPC 	Network Processin   
--	---	---

- ▲ Incorporated in 1996, spin-off of IMEC
- ▲ Independently owned, profitable company
- ▲ Worldwide reach

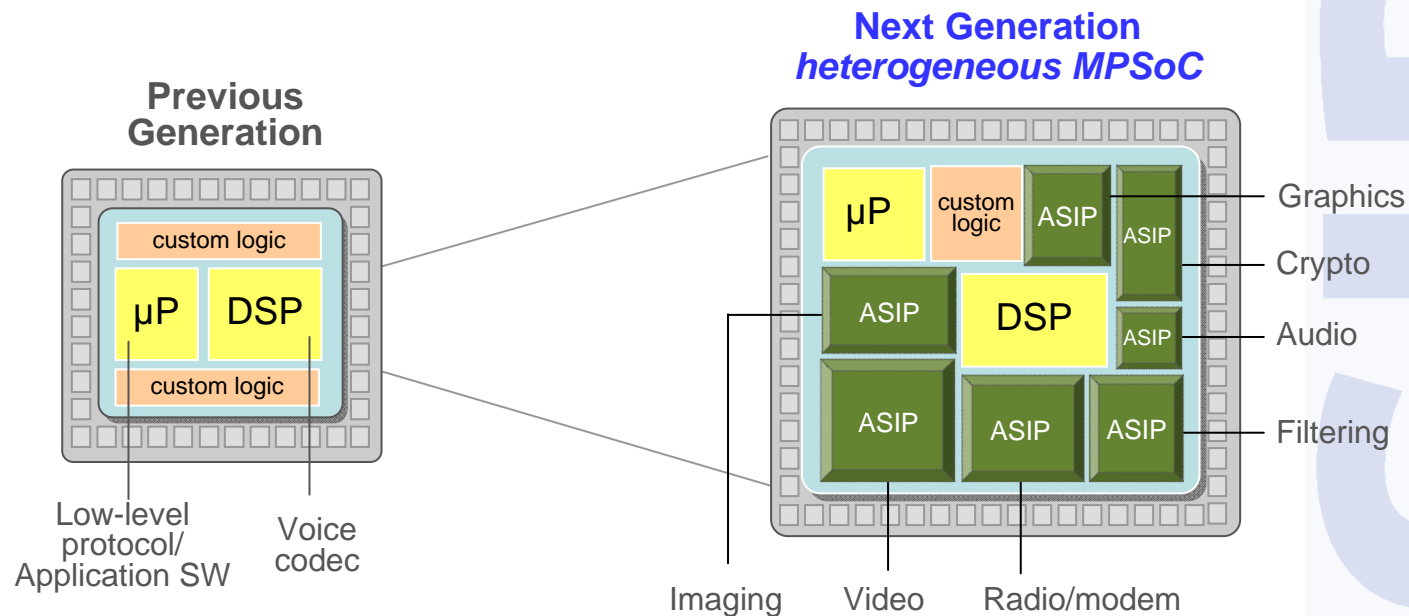
ASIPs – Application Specific Processors



▲ The Third Tool in the Box

- Anything between gen.-purpose microprocessor and hardwired data-path
- Customized in varying degrees to the application domain
- High throughput, low power through parallelism and specialization
- Programmable

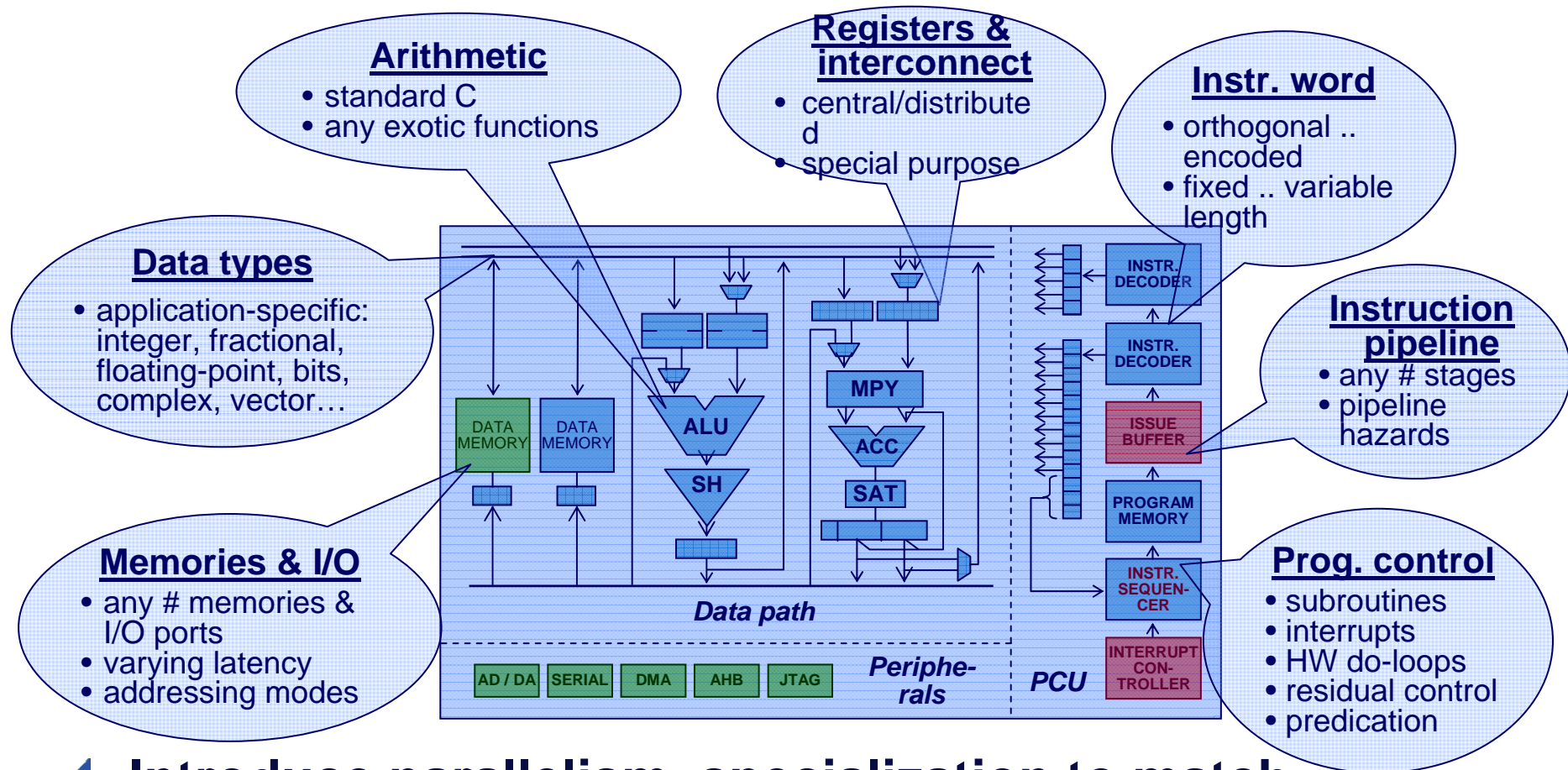
ASIPs enable MPSoCs



▲ Heterogeneous architectures are common

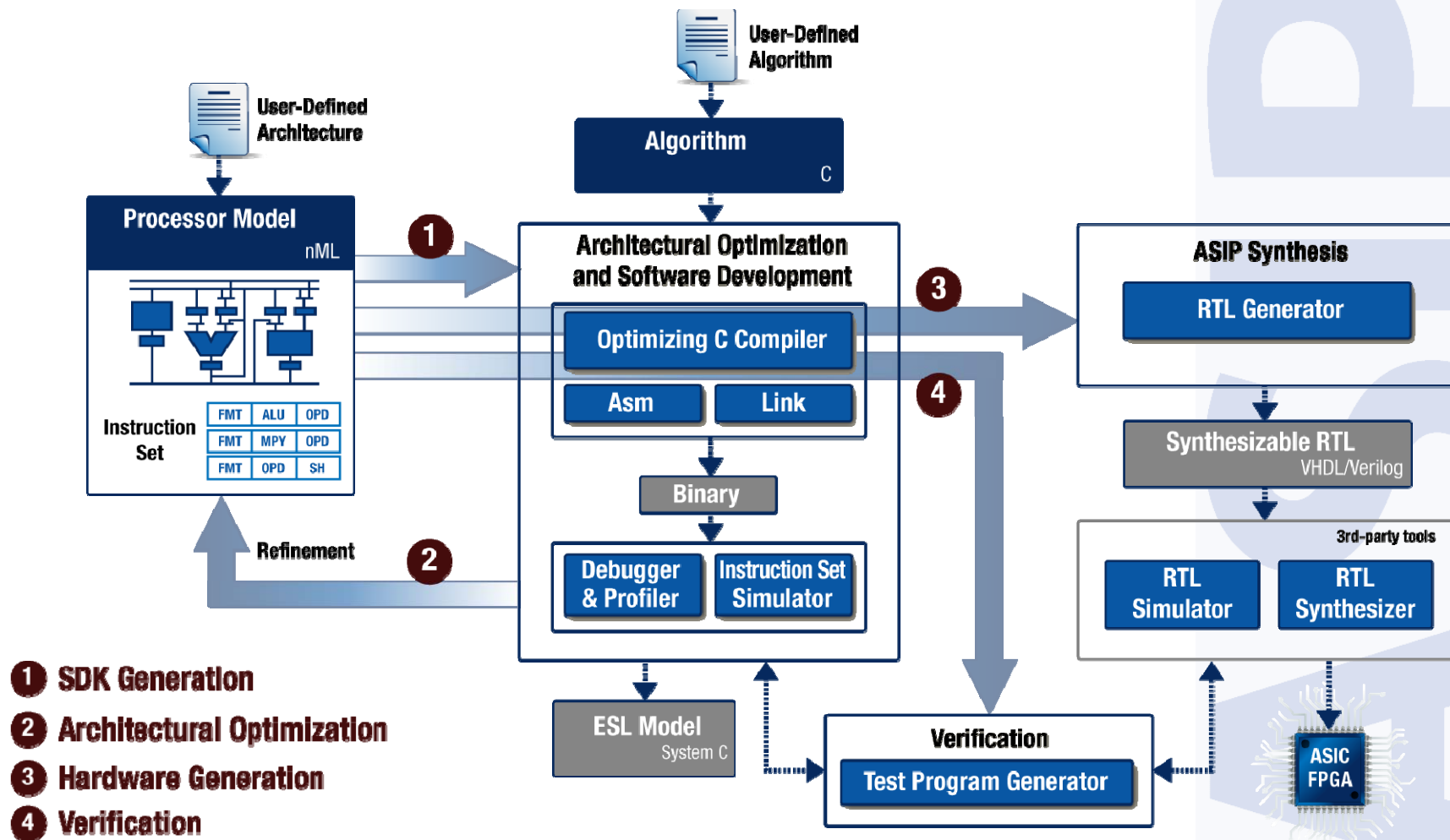
- Each ASIP optimised for its function: minimal logic, balanced parallelism
- Local communication
- Power gating based on system requirements

Architectural variation of ASIPs



▲ Introduce parallelism, specialization to match application's throughput and power requirements

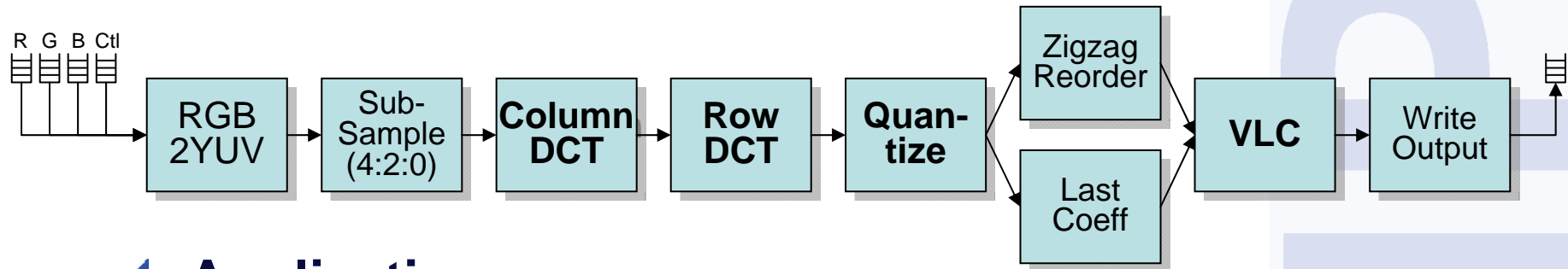
IP Designer Tool Suite



Agenda

- ▲ **ASIPs and Target – introductions**
- ▲ **High-throughput JPEG encoder design**
 - Specification
 - Multi-core partitioning
 - DCT ASIP
 - VLC ASIP
 - Evaluation
- ▲ **Conclusions**

High-Throughput JPEG Encoding



Application

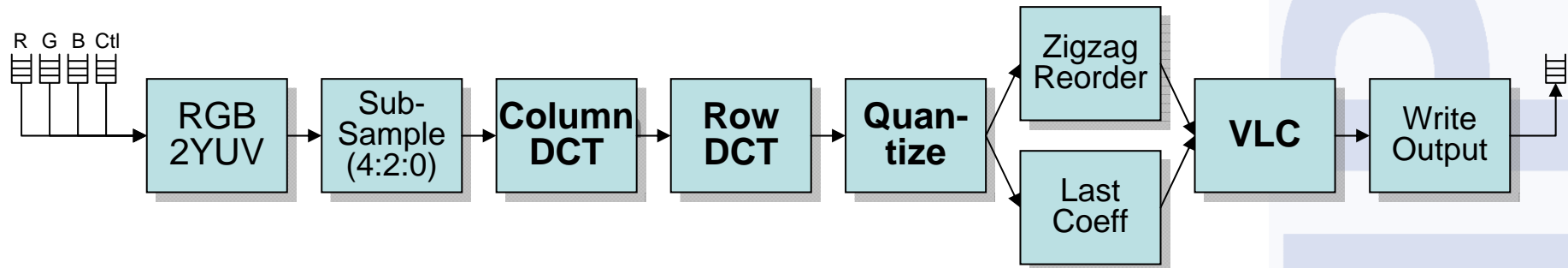
- Digital still camera, high-resolution digital scanner

Specifications

- Modified version of public-domain code [1]
- Throughput: 1 pixel / cycle (R+G+B / 3 cycles)
- Gate count: $\leq 100K$, excl. memories (100MHz, CMOS90)
Note: this is based on gate count of customer's existing fixed-function RTL plus some small margin for programmability
- Input & output FIFOs

[1] www.media-tool.com/guides/node/23

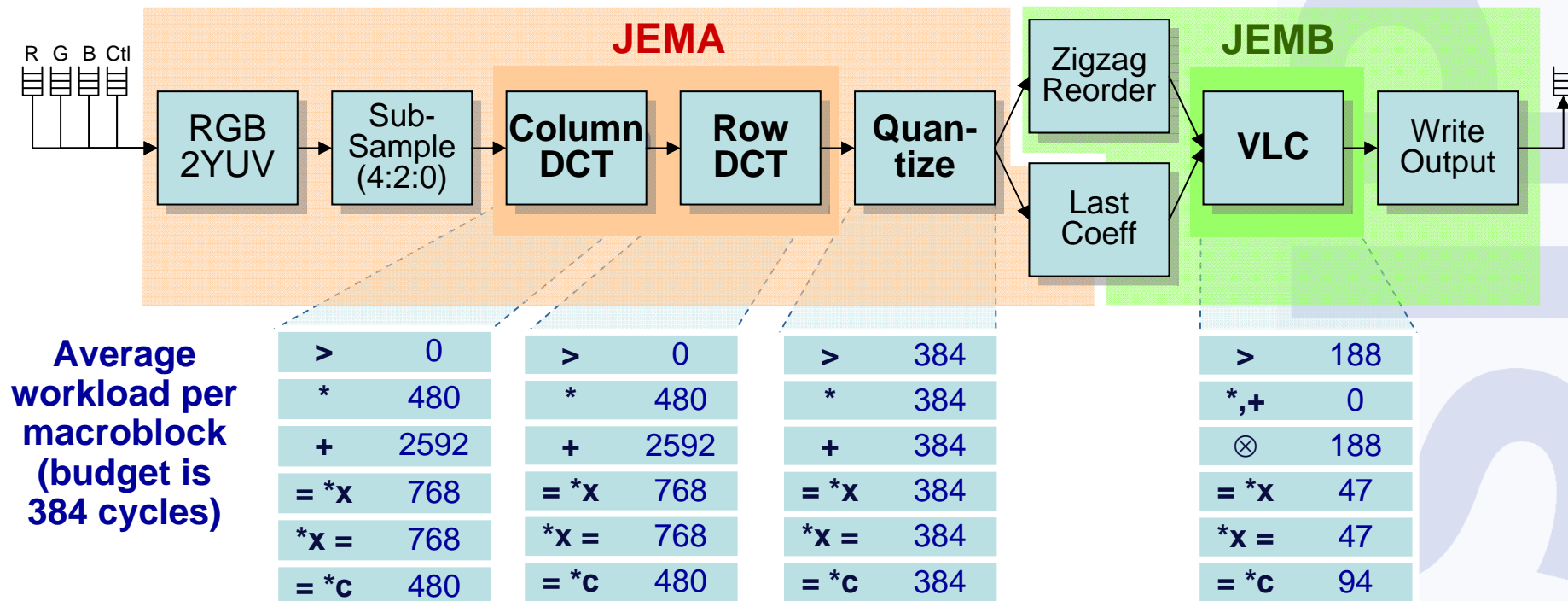
Multi-Core Partitioning



Initial partitioning inspired by system block diagram

- Map adjacent blocks to same core → local communication
- Map blocks with compatible data & control structure to same core → potential resource sharing
→ opportunity for ILP
- Map blocks with dissimilar control structure to separate cores → maximize HW utilization, pipeline parallelism
- Right-size each core, so that HW resources are used efficiently
→ max. performance/gate
→ min. clk freq and power
 - Data & instruction-level parallelism
 - Specialization

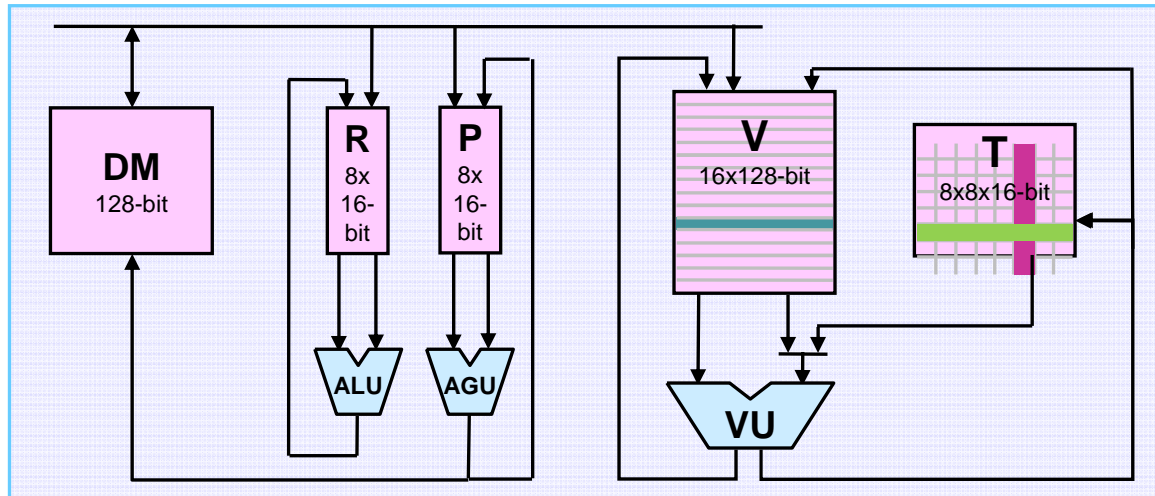
Multi-Core Partitioning



⊗ = VLC primitives

- DCT ↔ VLC: incompatible control & data-flow structure
- VLC workload feasible on scalar, specialized ASIP – “JEMB”
- Column DCT + Row DCT feasible on 8-element vector ASIP – “JEMA”
- Full partitioning fine-tuned using IP Designer

“JEMA” DCT ASIP – Initial



Architecture

- VU: basic vector arithmetic – vadd, vsub
quantisation and scaling primitives – vscale, vmul
- T: transposable register file – write columns, read rows

Performance

- Cycle count of compiled code: 150 cycles / DCT loop,
i.e. **150% over budget**

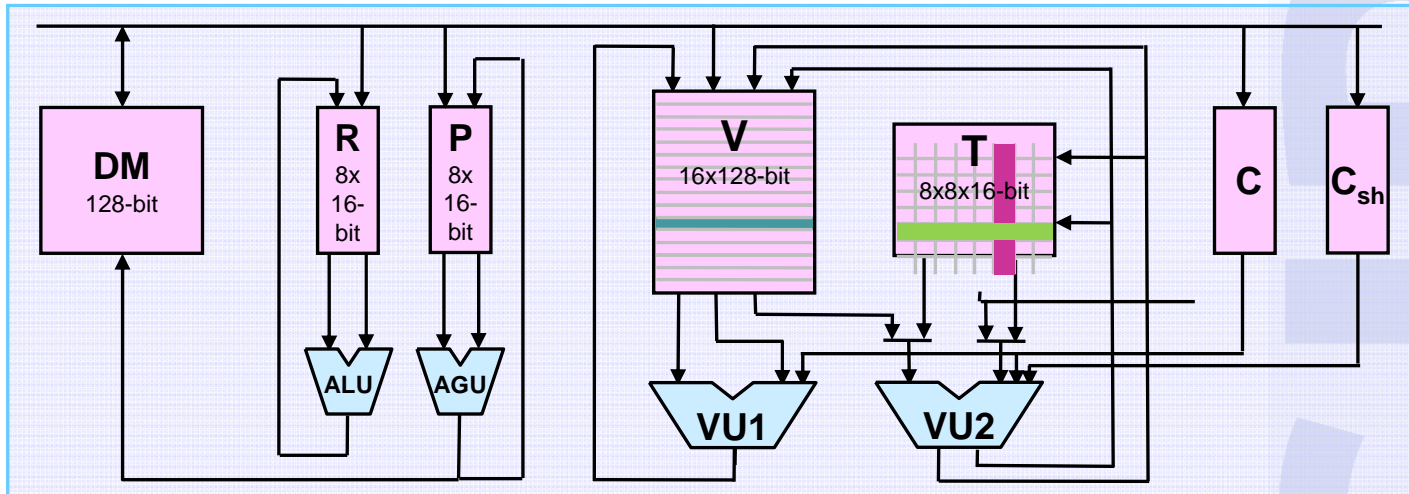
“JEMA” Initial - Observations

```
...
57 do r6,208
58 mvi r3,8
...
65 do r6,207
...
69 mvi r6,139
...
97 vmul v0,bcst(r6),bcst(r3)
...
118 vsub v2,v1,v7
119 vadd v6,v3,v5
120 vadd v8,v11,v0
...
123 vadd v1,v1,v7
124 vsub v3,v3,v5
125 vsub v0,v11,v0
...
127 trps_write v1, TC0
128 trps_write v2, TC4
129 trps_write v6, TC2
...
137 trps_read TR2, v3
138 trps_read TR3, v5
139 trps_read TR4, v7
...
207 ld r7,dm(sp-5)
208 nop
...
```

Assembly-level profiling

- No instruction-level parallelism
 - Add VU1 || VU2 instructions
 - Add load-store || VU instructions
- Frequent pattern: vadd, vsub with common operands
 - Add vaddsub instruction
- Multiple moves between T and V register files
 - Add T to operand/destination of VU instructions
 - Add dual read/write ports for T
- Limited size of R causes frequent reloading of constants for VU
 - Add constant register file

“JEMA” ASIP – Speed Optimization



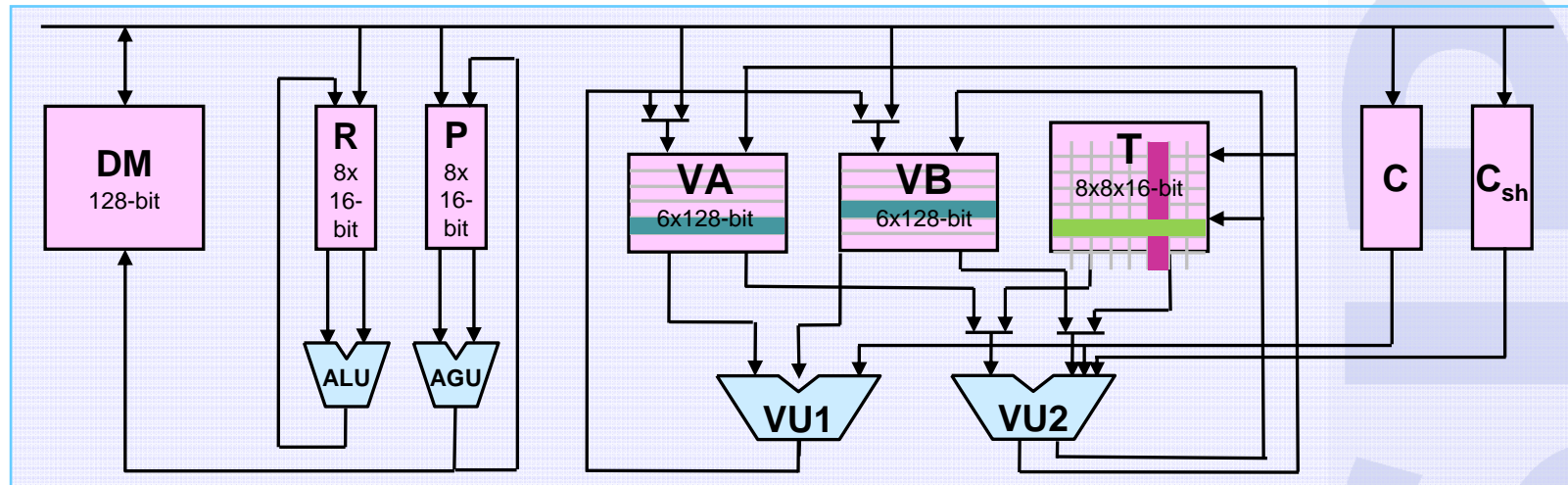
Architecture

- Added VU2 || VU1, load-store || VU2
- Added vaddsub, optimised distribution of VU1 & VU2 instructions
- Added dual T operands, constant register files

Performance

- Compiled code: 59 cycles / DCT loop, i.e. **within budget**
- Gate count: 90K, i.e. **almost entire JPEG encoder budget**

“JEMA” ASIP – Area Optimization



Architecture

- Split vector register file: V (16 elements, 4R + 4W ports) → VA + VB (6 elements, 2R + 2W ports each)
- Encoded opcode fields: 48-bit → 28-bit instruction word
- No changes needed to C code!

Performance

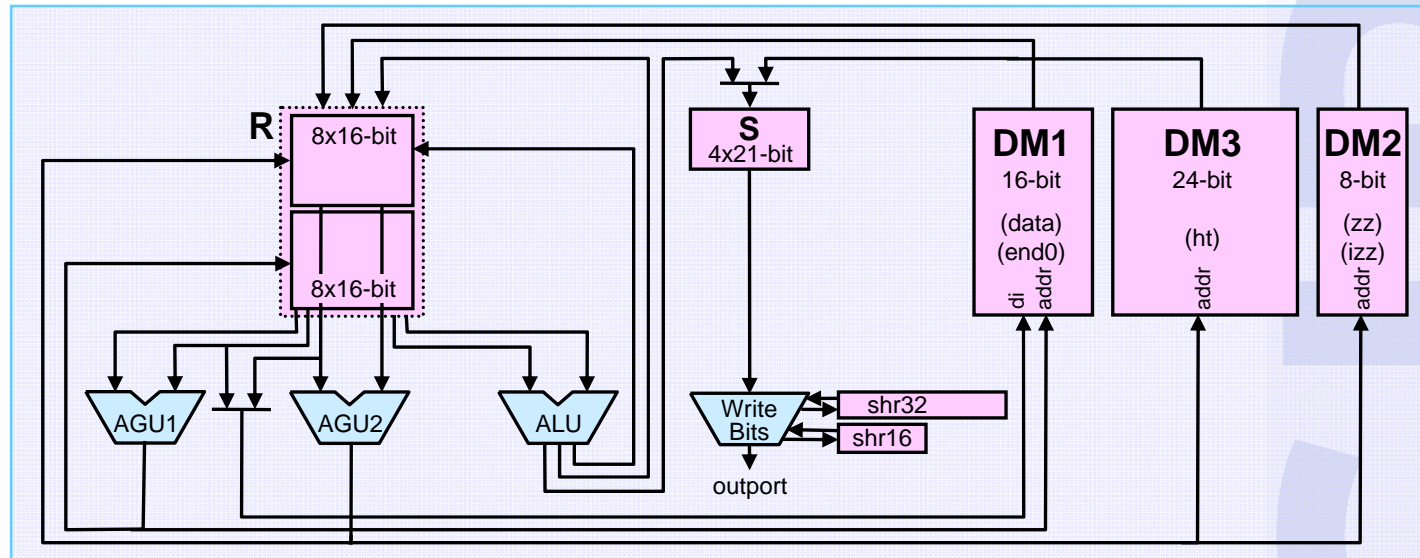
- Compiled code: 59 cycles / DCT loop, i.e. **within budget**
- Gate count: 65K, i.e. **within budget**

“JEMA” ASIP – Assembly Snippet

```
...
95 do r0,158
96 ld va0,dm(p0+=m1) || nop
97 ld va2,dm(p0+=m1) || nop
98 ld va1,dm(p0+=m1) || nop
99 ld va3,dm(p0+=m1) || nop
100 ld vb0,dm(p0+=m1) || nop
101 ld vb1,dm(p0+=m1) || ( add va3,va3,vb0 || sub vb0,va3,vb0 )
102 ld vb1,dm(p0+=m1) || ( add vb2,va1,vb1 || sub va1,va1,vb1 )
103 ld m3,dm(p1++) || ( add va2,va2,vb1 || sub vb1,va2,vb1 )
104 vadd vb4,va1,vb0 || ( add va4,va2,vb2 || sub vb2,va2,vb2 )
105 ld vb3,dm(p0+=m3) || vmove vb0,va3
106 vadd vb3,va1,vb1 || ( add vb5,va0,vb3 || sub va2,va0,vb3 )
107 vmove va0,vb5 || vmul vb3=bcst(c0)>>bcst(csh0)
108 vadd va0,va2,vb1 || ( add vb0,va0,vb0 || sub va1,va0,vb0 )
109 vsub va2,vb4,va0 || ( add va3,va2,vb3 || sub vb3,va2,vb3 )
110 vadd vb2,va1,vb2 || mul vb1=(va2*bcst(c1))>>bcst(csh0)
111 vmove va2,vb0 || vmac vb0=vb1+(va0*bcst(c3))>>bcst(csh0)
112 vmove vb0,va4 || ( add TC1,va3,vb0 || sub TC7,va3,vb0 )
113 vmove va0,vb4 || ( add TC0,va2,vb0 || sub TC4,va2,vb0 )
114 vmove va0,vb3 || vmac vb0=vb1+(va0*bcst(c2))>>bcst(csh0)
115 nop || mul vb1=(vb2*bcst(c0))>>bcst(csh0)
116 nop || ( add TC2,va1,vb1 || sub TC6,va1,vb1 )
117 nop || ( add TC5,va0,vb0 || sub TC3,va0,vb0 )
118 ld m3,dm(p5) || ( add vb0,TR0,TR7 || sub va0,TR0,TR7 )
119 vmove va4,vb0 || ( add va1,TR3,TR4 || sub vb0,TR3,TR4 )
120 ld p7,dm(p2++) || ( add va3,TR1,TR6 || sub vb1,TR1,TR6 )
121 vmove vb3,va1 || ( add vb2,TR2,TR5 || sub va2,TR2,TR5 )
...
```

- Portion of inner loop
- Target's C Compiler:
 - Schedules code
 - Picks VU1 vs. VU2
 - Assigns registers, incl. choosing VA vs. VB
 - Minimizes moves
 - Infers need for vaddsub
 - Uses T registers for transposition
 - Schedules address computations in parallel
 - Maximizes ILP

“JEMB” VLC ASIP



Architecture (after exploration with IP Designer)

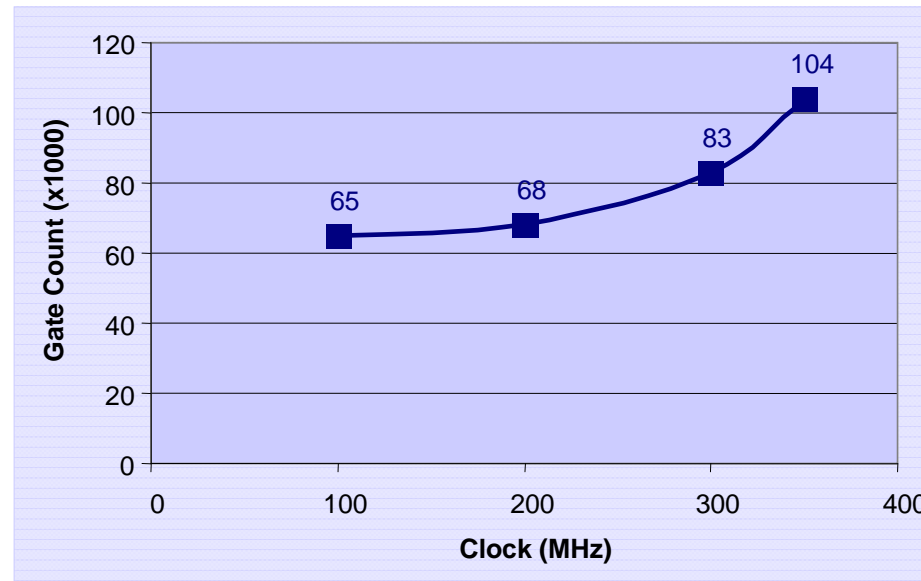
- Multiple data memories, for different data-types
- Instruction-level parallelism: DM1 || DM2 || ALU || WriteBits
- WriteBits: shift primitives for bit concatenation
- ALU: standard arithmetic + Huffman table address and code calculation primitives
- AGUs: mix of pre- and post-increment modes

Evaluation

▲ Dual-ASIP JPEG encoder architecture

- Gate count: 65K (JEMA) + 11K (JEMB) = 76K
- Throughput: 1 pixel / cycle

▲ JEMA RTL results (CMOS90)



*Significantly
below 100K
design
requirement*

Evaluation

▲ Effort: 1 person-month

▪ Preparing C source code	3 days
▪ Initial modelling of JEMA ASIP	3 days
▪ Architectural exploration* JEMA ASIP	4 days
▪ Initial modelling of JEMB ASIP	3 days
▪ Architectural exploration* JEMB ASIP	3 days
▪ Tuning partitioning	3 days
▪ Documentation	<u>4 days</u>
	23 days

* Tuning processor model, compilation, simulation, RTL generation and synthesis

Evaluation

▲ ASIP methodology offers HW flexibility

- JEMA ASIP initially intended for DCT only
- Added RGB→YUV and sub-sampling, with only minor HW extensions
 - R↔T moves, pair-wise intra-vector addition

▲ ASIP methodology offers SW flexibility

- Added control code for .jpg-file headers and markers to JEMB
- Can add calculation of quantisation coefficients to JEMA

Conclusions

▲ Dual-ASIP JPEG encoder design

- Very short design time
- Meets throughput and beats gate-count specs
- Highly flexible solution

▲ IP Designer is a key enabler

- Highly-parallel, finely-tuned architectures
- System-level partitioning
- Rapid architectural optimisation

▲ The ASIP Advantage

- Reconcile flexibility ↔ high throughput ↔ low power
- Bridge gap from C to RTL, in a controlled way