

PVP

a Parallel Video coProcessor

France Telecom CNET

Dominique BARTHEL

VLSI design manager

dominique.barthel@cnet.francetelecom.fr

a Parallel Video Processor

Application background

Telecom company and TV broadcaster

- contribution to standardization committees
- building real time prototypes (multimedia, digital TV encoders, HDTV)

Many advanced ASICs designed for these environments

- filters
- 8K FFT, DVBird chips

Looking for a “generic” component for video

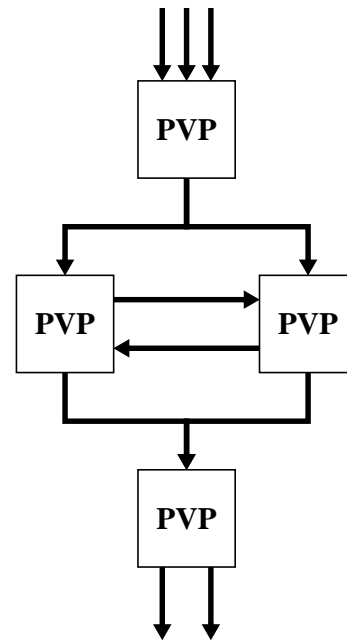
- programmable
- allowing easy system integration

System Perspective

Not just a powerful chip

Also allow a scalable MP system

- **total bandwidth increases with number of PVPs**
 - 7 unidirectional busses per PVP
 - IO fully asynchronous to core
- **total memory increases ...**
 - each PVP has internal data memory
 - data memory may be expanded outside PVPs through a DMA mechanism
- **easy clock distribution**
 - no phase/frequency constraints between PVPs



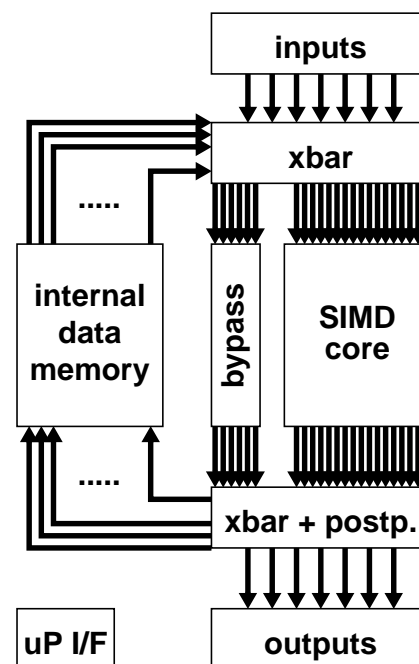
Chip architecture

SIMD core with 16 PEs

Internal data memory (KBytes) allows 16 reads and 16 writes per cycle

Data streaming architecture

- operands come from inputs or memory or internal registers
- results go to internal registers or memory or outputs

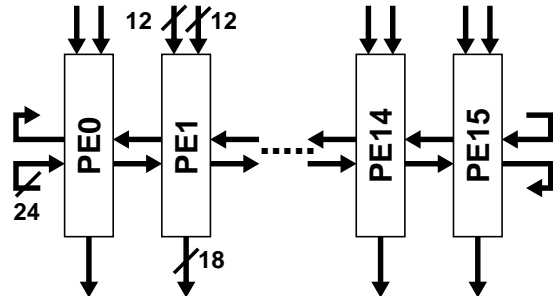


SIMD core

SIMD structure

- 2 operands per cycle, 1 result
- individual register files
- PEs can operate conditionally
- ring topology
 - shift right, shift left or butterflies

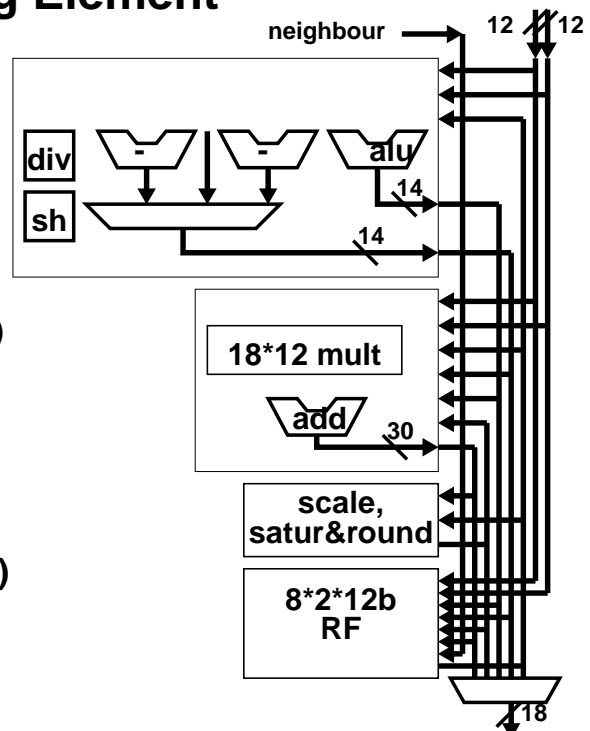
Data widths optimized for video



Processing Element

A pool of operators

- 1 ALU
- 2 SUB
- 1 shifter
- HW assist for division (2 bits/cycle)
- 1 multiplier
- 1 adder
- 1 scale, round and saturate logic
- 8 pairs of 12 bit registers
- 6 flags (conditional operation, ALU)
- exposed pipeline (no scoreboard)



Memory (1)

Memory of 8 bit or 12 bit words

- specified at boot time
- first implementation has 16K words of 8 bits, or 8K words of 12 bits

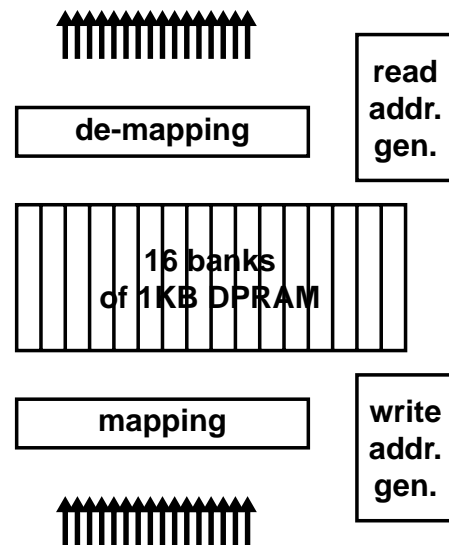
Logically seen as a 2D memory

- X and Y dimensions defined at boot

16 reads and 16 writes per cycle

- 16 banks of Dual Ported RAM
- conflicts must be statically resolved (i.e. avoided)

Addresses update every cycle



Memory(2)

Avoiding bank conflicts

- usual SIMDs only allow aligned packets of consecutive bytes
- in PVP, skewing function defines legal sets of logical addresses
- skewing function defined at boot

Supplying 32 addresses per cycle

- 16 addresses defined as
 - an origin
 - a pattern relative to the origin
- every cycle, update of the origin (2D marching law) and of the pattern

8 address generators available

128 pre-recorded patterns available

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 |
| 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 |
| 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 |
| 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 |
| 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 |
| 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 |
| 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

mapping example
for 8 banks

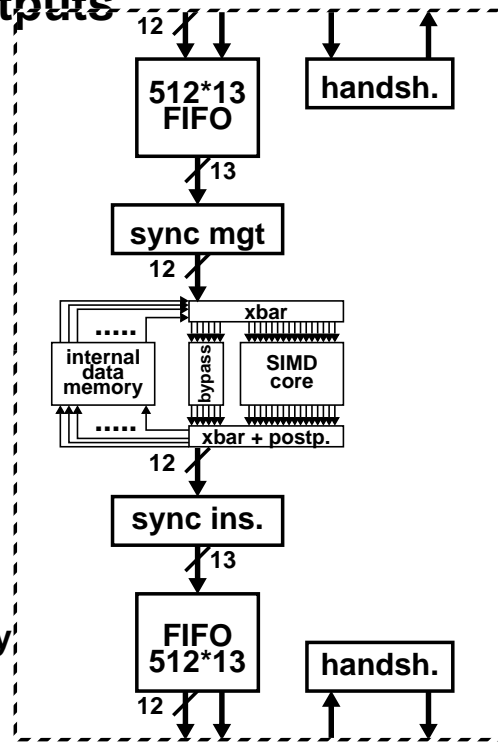
Input/Outputs

Ease system development

- handshaking taken care of by hardware
- FIFOs even out bursts
- IO clock asynchronous to core
- blocking load/store

Allow easy synchronization

- “separator” tokens for synch
- hierarchy of separators
- separator read/write instructions
- semi-automatic input synch. recovery



Instruction set

Width vs. flexibility trade-off

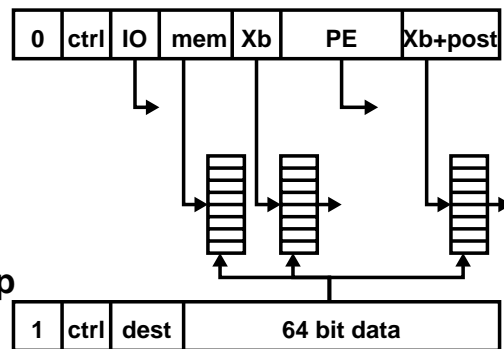
- 112 bit VLIW
- direct control of the SIMD PEs
- mem, Xbars need 100's of bits!
- pre-recorded tables to hold configurations relevant to current app

Instructions

- few bits to many operators
- some fields index into tables

Directives

- many bits to one target
- load tables, control registers and memory with constants



```
REPEAT | RW DF0 | RD MEM 2 | XBAR 4 | LD MULT E1,E2
| SEL REG HL1 | ADD M,A | LD SPOUT,G
```

```
DIR LOOP L234 = 64 8 PREV
```

Flow Control

Co-processor, not a full blown uP!

- executes only straight line code, or fixed trip-count loops
- loops nested up to 5 deep, plus 1 REPEATED instruction
- external uP sends Program Entry Points
- first implementation holds 1K VLIWs in internal prog memory

One special IO to exchange data with external uP

- 128-deep FIFO
- values supplied by uP can be used as “immediates” in directives

Pipeline

Time-stationary instructions

- uses many instructions for pipeline fill and drain
- needed when no “natural” path through operators

Data-stationary instructions

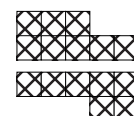
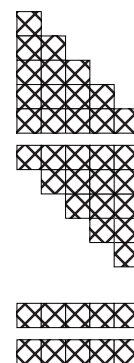
- controls travels down the pipeline with the data
- the “travel” must be well defined

PVP has 21 pipestages

- most of them through a “natural” pipeline (fetch, decode, addr, mem)
- the PE operators are not naturally ordered

Hybrid model used

- data stationary outside the PE
- time stationary inside the PE



Conclusions

First implementation under way (July 97)

- 0.5 um 3.3V, 70 MHz
- 8 PEs
- 700 MB/s IO
- 3 GOPS

Next implementation 0.25 um early 98

Good flexibility/performance trade-off

Easy prototyping of algorithms/applications

New application fields (xDSL)

Open for collaboration/licensing