Reflections on PPF: New directions in constructing high-specification image processing and multimedia embedded software
Martin Fleury and Andy Downton

Talk outline

• PPF project — processor parallelism in the era of modular parallelism.
• Some recent projects in large-scale signal processing.
• Embedded systems — hardware and software challenges — beyond modular parallelism.
• RaPPID project.
• MiPPS project.
• Conclusions.

PPF

• Pipelined Processor Farms (PPF) originated as a simple, unifying idea connecting data-farms in a single pipeline — processor-level parallelism.
• The data-farm supports dynamic scheduling and incremental scalability.
• The pipeline allows sequential bottlenecks to be masked, thus escaping a consequence of Amdahl's law (the performance is effectively controlled by the sequential content).
• PPF has an associated development cycle — transformation of serial code (including legacy code) to parallel form, without changing the algorithms.

PPF (continued)

• PPF was first devised in the era of optimism on the future of the modular multicomputer, as a way of countering the effects of Moore's law.
• Transputer is the ancestor of the 'glueless' multiprocessor, other examples: the TI 'C40, Analogue Devices SHARC DSP series — medium-grained parallelism.
• Data-farm on a chip — TI 'C80 suggests hardware extension to PPF.
TI 'C80 (MVP)

- Backplane can be high-bandwidth bus — communication scalability.
- Regular topology — permits replaceable data-farm nodes.
- Scheduling synchronised by returning work (or tokens).
- Local and inter-stage buffering smoothes data-flow.

Classic PPF
Cascaded data farms

- Low-latency solution — deterministic workload.
- Data-farmer bottleneck reduced.
- Data-flow synchronisation if semi-deterministic processing cycle.
- Irregular topology, no communication hierarchy.

Hardware for PPF

- Paramid multicomputer combines i860 vector-processor computation engine with transputer coprocessor — communication via overlapped memory.
- (Compare Meiko CS2 combines SPARC, Elan coprocessor, and 2 Fujitsu μ VP vector processor units)
- Communications coprocessor allows test and monitoring framework to be built in to template.
• Data-farm template used to construct prototype pipelines.
• Paramid is as much development/demonstrator environment as it is target hardware.

Asynchronous pipeline example: Postcode recognition

• i860 version with transputer comms. — in the transputer version the no. of processors was an order of magnitude greater.
• Granularity controlled by the application the dictionary search is parallelisable but not for this pipeline and for these processors.

Synchronous pipeline example: H261 encoder

Synchronous pipeline: H261 encoder (continued)
<table>
<thead>
<tr>
<th>Execution sequence</th>
<th>Function name</th>
<th>Per frame exec. time (normalized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>clear_picture</td>
<td>0.147</td>
</tr>
<tr>
<td>2</td>
<td>copy_field</td>
<td>0.196</td>
</tr>
<tr>
<td>3</td>
<td>read_picture_frame</td>
<td>1.200</td>
</tr>
<tr>
<td>4</td>
<td>copy_macro_block_data</td>
<td>0.728</td>
</tr>
<tr>
<td>5</td>
<td>encoder_motion_estimation_h261</td>
<td>24.306</td>
</tr>
<tr>
<td>6</td>
<td>encoder_decisions_h261</td>
<td>6.392</td>
</tr>
<tr>
<td>7</td>
<td>forward_transform_picture</td>
<td>2.230</td>
</tr>
<tr>
<td>8</td>
<td>h261_forward_quantize_picture</td>
<td>1.398</td>
</tr>
<tr>
<td>9</td>
<td>tm_forward_scan_picture</td>
<td>0.632</td>
</tr>
<tr>
<td>10</td>
<td>forward_run_level_picture</td>
<td>0.712</td>
</tr>
<tr>
<td>11</td>
<td>h261_code_picture</td>
<td>0.591</td>
</tr>
<tr>
<td>12</td>
<td>inverse_run_level_picture</td>
<td>0.486</td>
</tr>
<tr>
<td>13</td>
<td>tm_inverse_scan_change_picture</td>
<td>0.634</td>
</tr>
<tr>
<td>14</td>
<td>h261_inverse_quantize_picture</td>
<td>0.991</td>
</tr>
<tr>
<td>15</td>
<td>inverse_transform_picture</td>
<td>2.232</td>
</tr>
<tr>
<td>16</td>
<td>reconstruct_h261</td>
<td>2.813</td>
</tr>
<tr>
<td>17</td>
<td>macro_block_to_line</td>
<td>0.531</td>
</tr>
<tr>
<td>18</td>
<td>tm_calculate_snr</td>
<td>0.753</td>
</tr>
<tr>
<td>19</td>
<td>tm_display_statistics</td>
<td>0.134</td>
</tr>
<tr>
<td>20</td>
<td>write_picture_file</td>
<td>1.217</td>
</tr>
</tbody>
</table>

**Synchronous pipeline: H261 encoder (continued)**

**PPF characteristics**

- Continuous flow, data-driven (not control-driven), soft real-time with possible throughput, latency, & ordering constraints accommodated.
- Aimed at software-based systems with homogeneous processors.
• Suitable for multi-algorithm systems with upwards of 5,000 lines of code.
• Limited number of feedback loops — folded-back pipeline possible.
• Performance models (dynamic) for synchronous and asynchronous pipelines have been developed — linear programming, order statistics.
• Farm is a general concept, though in practice only temporal and/or data farms are used — other algorithms are handled centrally.

Some categories of application where PPF is inappropriate

• Cellular-array algorithms relaxation algorithms in optical-flow algorithms — suitable for target tracking. Problem: suitable SIMD hardware not sustained in the marketplace.
• Fine-grained synchronous algorithms beam-forming of adaptive- or delay-filter variety.
• Large, shared data-structure as in HMM (Hidden Markov Model) speech-recognition model — small-scale Symmetric Multiprocessors (SMPs) are becoming common due to the database market and suit this model — may also require message-passing parallel phase.

'Ill-conditioned' pipeline
'Ill-conditioned' pipeline (continued)

- Large data bandwidth and compute/communicate imbalance – possibly ameliorated by shared-memory parallelism and multithreading.
- Large data buffering requirements (image sets) if the pipeline is operated in continuous flow.
- On general-purpose processors, the Eigenvector calculation on a small dimensional matrix is relatively fast compared to the formation of the covariance matrix.
- Strict synchronisation across the pipeline stages.

'ILL-conditioned' pipeline (continued)

- Multiple farms increase the number of processors to effect scaling, and again there is an imbalance in the three stage pipeline.
- First farm is suitable for SIMD implementation, as is the third farm though requiring real-valued calculations — in fact, the algorithm has been parallelised on the DAP.
- RISC/systolic or RISC/FPGA solution may restore balance — if large i/o bandwidth.
- Heterogeneous hardware raises the possibility of using slower, but cheaper processors in shorter duration stages.

Tracee project extension
• Light-weight version of Khoros visual-programming environment developed at Kings and used within DERA, GEC-Marconi
• Suitable for low-latency version of PPF — data-farm allows communication/computation overlap.
• Cascaded data-farms model.
• Processor nodes may be host with FPGA accelerator.
• Introduces software/hardware partitioning; FPGA targets may be reconfigurable/adaptive.
• Preserves integrity of code originating on the developer’s workstation/PC.

**Example: Processing problems facing radar**

• Sample rates now approaching 500MHz.
• Multidimensional, and nonlinear signal processing leads to:
  * 2D & 3D imaging of terrain or objects — Synthetic Aperture Radar (SAR).
  * Estimation of movement and velocity of moving targets — not only target-tracking.

• Algorithmic changes are endemic, though some staple algorithms, notably FFT.
• Real-time processing of data within the sample time-frame especially for military applications.
• Some special requirements: I/O bandwidth may be critical; aircraft will require compact, low-power solution.

**Two-level computer for Automatic Target Recognition (ATR)**

![Diagram of Two-level computer for ATR]

**Two-level computer (continued)**

• Synthetic Aperture Radar (SAR) — data capture and pre-processing of images.
• Focus of Attention (FOA) — extracts regions of interest within images with potential targets (moving vehicles).
• Second-level detection — simple time-domain templating matching (bright mask and surround mask)— embarrassingly parallel.
• ORCA FPGA solution investigated by Myricom.
• 'Evolving algorithms and high non-recurring engineering costs limit usefulness of ASICs.'

Two-level computer (continued)

- First level is LANai microprocessor NIC for network interfacing, message handling, initialisation, FPGA (40K gates).
- Four FPGA nodes are connected in local System Area Network (SAN) by 8-way crossbar. (FPGAs can also be connected locally in a ring topology)
- Higher-level connectivity is via Myrinet LAN connection (160 Mbytes/s 1998) — replicates interconnect common on Cray T3D and other MPPs.
Tower of Power (TOP)

• Virginia Tech. offering has a similar architecture to Myricom ATR, though perhaps at a lower cost.
• 16 nodes with Pentium II hosts (300MHz) connected by Myrinet switch hierarchy. (PowerPc family are preferred host solution, because of cleaner architecture?).
• Pentiums interface to Wildforce board with 5 FPGAs (Xilinx 4062 parts).
• Again FPGAs connect in ring topology.

Sonar beam-forming solution (A)

• Passive sonar for target tracking & identification (SLAAC), two systems at B. Y. University:
  1. Linear array of 400+ sensors mounted on submarine, 1-2KHz sample rate, 12-bit data, 13,000 beams resulting in angle, range, and frequency content.
  2. Spherical array 600 beams, 25KHz sample rate.

• Similar hardware to TOP is proposed for SIMD operation (delay or frequency):
  • 1 FPGA (Xilinx XC4028XL-1 c. $300) equivalent to 12 SHARCs (ADSP21060 40 MHz $325) — Linear array.
  • 1 FPGA (Xilinx XC4052XL-1 c. $1000) equivalent to 6 SHARCs (Shared memory configuration) — Spherical array.

• DSP bottleneck is (apparently) memory access due to irregular addressing pattern for these algorithms.
• FPGA bottleneck will be access i/o pins.
• Adaptive beam-forming will require synchronous broadcast of coefficients as well as samples.

Sonar beam-forming solutions (B)

• University of Texas propose software solution on commodity SMP for interpolated time-delay beamforming.
• SMPs, Sun Ultra Enterprise 12 processor 336MHz, promise a four-fold reduction in development costs/time — Speedup 11, c. 4.8 GFLOP/S.
• 80 10 sensor array, beamformer produces 3 sets of vertical responses from which 61 sets of horizontal beams are produced for each set.
• Visual Instruction Set (VIS) integer, word-level SIMD (vertical beam), optimised (loop unrolling) C++ utilised, with added data pre-fetching, (f.p. horizontal beams).
• Pthread implementation guarantees portability — question-mark over determinism —
data-flow programming paradigm works irrespective of scheduling regime (?) — Kahn process networks.

Sonar comparison

• DSP/FPGA comparison reveals need for careful benchmarking — may impede development of performance model.
• SMP solution is essentially software-based, requires detailed knowledge of S/W optimisation data pre-fetches, while DSP, FPGA requires detailed knowledge of H/W, use of CORDIC for magnitude/phase manipulations.
• DSP clock speeds remain comparatively low. Low-cost Xilinx FPGAs also peak 50 MHz.
• Disparity in number of beams formed, but more complex processing on SMP.

Prospective hardware

• Fine-grained hardware now available at the VLSI level:
  • Virtex family FPGA have 1M gates, no premium on floor space:
    • silicon compilation avoids special-purpose algorithms.
    • possibility of going from FPGA to purely software solution — software-synthesis.
    • hardware parallelism greater than DSP (FFT engines).
    • i/o does not scale with gate numbers.
    • high power consumption (25W at 100MHz).
    • requires non-volatile FPGAs for faster switching speeds.

• RISC/FPGA combinations NAPA1000, Chameleon System 2112.
• RISC/SIMD combinations DIP IC from Sheffield — RISC controller and PE array acting in SIMD mode or pipeline of rows; Micron also have plans for a DAP on a chip.

Prospective hardware (continued)

• Internal parallelism, VLIW processor, superscalar commodity processors, out-of-order execution (dataflow), instruction predication — makes massive parallel processing less likely in the medium term — processor nodes are simply too powerful.
• Store-and-forward communication always limited scalability.
• SHARC DSP family — limited form of scalability together with support for shared-memory parallelism.
• ASICs should be considered for standard algorithms with regular structure DCT, FFT — Plessey FFT and complex multipliers used in Pharus radar pulse compressor.
Hardware obsolescence

- Example: a typical commercial microprocessor has a product lifespan of less than five years lifespan, while a typical aircraft has a product lifespan of over five years.
- Equally, a company will have a number of clients with different target systems and different specifications.
- Algorithm developers are well aware of the danger of fixing their software to an implementation, and consequently are reluctant to move away from PC/workstation development environment.
- Result: separate implementation phase with errors arising when software is partitioned.
- Software may have format weaknesses, pointers, or structural weaknesses, excessive reliance on feedback.

Overcoming obsolescence

- Core version of software with standardised structure.
- Sequential software is developed with minimal additional constraints to ensure parallel codesign is a trivial extension.
- High-level performance projection onto differing target platforms.
- PCs and workstations slow down algorithmic testing and design iteration for large-scale applications.
- An extension of the parallel development environment is required either SMP or NOW.

Software components

- Component model — extends object model: by larger-scale objects, and by incorporating dynamic structure — inheritance is optional.
- Data-farm templates demonstrated the utility of building in standard communication patterns.
- The data-farm template can be extended to other common parallel programming paradigms.
- Two-tiered organisation:
  1. control superstructure into which application functions are slotted.
  2. communication/coordination infrastructure.

Software components (continued)

- Performance monitoring instrumentation can also be built in — if overhead is low then there is no need to turn monitoring off — but 'probe effect' can occur.
- Granularity of task governs performance prediction — two-parameter model may be sufficient for top level estimate.
- Prototyping environment requires test data, record of performance runs, objective testing regime.

Other parallel paradigms
**Data-farms** each process works on a task to completion controlled by data farmer.

**Task-queue** each process works upon a task and then places the task on a queue for further processing by another process.

**OR-parallelism** each process performs differing algorithms on the same data or performs the same algorithm from differing starting points.

**Geometric** simple partitioning of data.

**Divide-and-conquer** recursive division of work amongst processes, results recombined in successive phases.

**Relaxation** successive synchronised phases with intervening local communication between PEs.

### Other parallel paradigms (continued)

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Data locality</th>
<th>Load-balancing</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data farm</td>
<td>Strict locality</td>
<td>Dynamic</td>
<td>Spatial convolutions</td>
</tr>
<tr>
<td>Task queue</td>
<td>Relaxed locality</td>
<td>Dynamic</td>
<td>Speech recognition</td>
</tr>
<tr>
<td>OR-parallelism</td>
<td>Strict locality</td>
<td>Semi-dynamic</td>
<td>Pattern matching</td>
</tr>
<tr>
<td>Geometric</td>
<td>Strict locality</td>
<td>Static</td>
<td>Image histogram</td>
</tr>
<tr>
<td>Divide-and-conquer</td>
<td>Relaxed locality</td>
<td>Static</td>
<td>Orthogonal transforms</td>
</tr>
<tr>
<td>Relaxation</td>
<td>Relaxed locality</td>
<td>Static phases</td>
<td>Image enhancement</td>
</tr>
</tbody>
</table>
Processing node

Software model
RISC, DSP

Software component

Control
Application functions

f1

f2

User space
System space
RTOS

API
IPC

NIC
Comms. S/W
Comms. H/W

Control
Application functions

f3

NIC
Comms. S/W
Comms. H/W
What is RaPPID?

- 3-year programme starting in October 2000 in association with DERA, Malvern.
- Algorithmic development on Networks of Workstations (NOWs) and/or Symmetric Multiprocessors (SMPs).
- A set of software components capture parallel structure but the granularity can be varied.
- Software components have built-in performance estimation, and the programming environment has a test-run toolkit.
- Low-level implementation on FPGAs, DSPs (SHARCsi glueless multiprocessor), or some combination of RISC and FPGA.
- Early interest is in the Handel-C silicon compiler from Embedded Solutions Ltd., Abingdon — educational usage is also possible.

How RaPPID goes beyond PPF

- Software components retain prescriptive style of PPF—simplistic approach single-steps engineer through the design process.
- Data-farm pipeline stages are replaced by coprocessors — coprocessor is the physical embodiment of software component.
- Coprocessor gives greater flexibility in solving pipeline bottleneck problem with heterogeneous hardware.
- Therefore, the new environment requires more ways to integrate hardware into a design.
- Equally, it may be necessary to extract a software solution from a design originally intended for hardware.
What is MiPPS?

• Many multimedia domain applications have two-phased structure:
  1. Data-reduction or pre-processing Gaussian mixture model calculations, edge detection.
  2. High-level processing speech recognition n/w update, component identification.

• 2nd phase requires access to common data-structure — multi-threading on SMP.
• 1st phase requires localised processing — emulation of message-passing on SMP by memory partitioning.
• MiPPS develops parallel programming environment/API within an object-oriented framework.

Conclusions

• Hardware diversity (especially in multimedia domain — less so in large-scale signal processing) is causing a rethink on long-term provision for embedded systems.
• Present balance between DSP, FPGA, and SMP.
• Two-level computer needs to be complemented by stable software structures.
• Software synthesis maybe the key to flexibility.
• Software component model takes on board the needs of developers as well as the target system.
• Performance models are the other requirement, but these can be built into the component model.
• High-level models are insufficient in themselves — h/w, s/w tweaking makes a vital performance difference.