21.11 APPLICATION AREAS FOR XPUTERS

Xputers like the MoM-3 and MoM-4 architectures are as universal as computers. A general competition between xputers and computers in all possible application areas would be unrealistic. This section briefly discusses suitable application areas for xputers from different points of view: for which algorithms and problem areas most benefits are to be expected - from which application environments least technology transfer problems will arise - in which application scenarios most cost/performance benefits can be expected. 330

21.11.1 Generalization of systolizable algorithms

In section 22.6.1 close relations of some compilation techniques for xputers to ASAP synthesis have been shown (ASAP stands for Application-specific Processor Arrays, like e. g. systolic arrays). Adapted versions of projection methods derive data schedules from data dependencies with the goal, that the right data is found at the right memory location at the right time throughout the whole execution of the algorithm. From the ASAP point of view an Xputer looks like an ASAP with only a single PE. The only difference is, that not the data are streaming, but scan cache locations are moving according to a scan patterns. Since a single processing element does not have neighbors, there are not the locality restrictions (unavoidable for ASAPs, where PEs can communicate only with their nearest neighbors).

This means, that with xputers the performance benefits of data dependence analysis could be extended to a much larger class of algorithms: to non-systolizable algorithms with regular data dependencies. We call the consequence of this extension of systolization methods a *generalization of systolizable algorithms.* Address generators may have such powerful repertories of generic scan patterns [56], that such algorithms are efficiently supported by hardware. Encouraged by performance result having been obtained experimentally we expect substantial superiority of xputers over computers in application areas like image processing, digital signal processing, computer graphics, multi media applications, scientific computing and others.

21.11.2 Application Environments

General purpose DS and image processor. Xputers are not competitive to computers in general, since cross compilers, and application software environments are not available commercially. Competitiveness, however, is be expected for particular niches of application markets, such as image processing, digital signal processing, computer graphics, multi media applications, scientific computing, and others, where higher performance is needed at low hardware cost.

General purpose accelerator. From a technology transfer point o view, and, for utilization of existing utilities, interfaces and application software an good symbiosis would be using the xputer as a universal accelerator co-processor, hosted by a von Neumann computer, e. g. as an extension board within a workstation. Only those critical algorithms, which exceed the power of the host, are candidates for running on the co-processor, mostly only a few lines of source code. 331 332

```
Array          PixMap [1:8,1:8,15:0]                                  (37)
ScanPattern    EastScan      is  1 step  [ 1, 0];                     (38)
               SouthScan     is  1 step  [ 0, 1];                     (39)
               SouthWestScan is  7 steps [-1, 1];                     (40)
               NorthEastScan is  7 steps [ 1,-1];                     (41)
                                                                      (42)
               UpLzigzagScan    is                                    (43)
               begin                                                  (44)
                    while (@[<8,])                                    (45)
                    begin  Eastscan;                                  (46)
                           SouthWestScan until @[≤1,];                (47)
                           SouthScan;                                 (48)
                           NorthEastScan until @[,≤1];                (49)
                    end)                                              (50)
               end UpLzigzagScan ;                                    (51)
                                                                      (52)
               LoRzigzagScan    is                                    (53)
               begin                                                  (54)
                    while (! @[8,8])                                  (55)
                    begin  EastScan;                                  (56)
                           NorthEastScan until @[8,];                 (57)
                           Eastscan;                                  (58)
                           SouthWestScan until @[,8];                 (59)
                    end                                               (60)
               end LoRzigzagScan ;                                    (61)
                                                                      (62)
               JPEGzigzagScan    is                                   (63)
               begin                                                  (64)
                           UpLzigzagScan                              (65)
                           SouthWestScan;                             (66)
                           LoRzigzagScan                              (67)
               end JPEGzigzagScan ;                                   (68)
endScanPattern ;        (* end of declaration part*)                 (69)
                        •                                             (70)
                        •                                             (71)
                        •
begin                   (* statement part*)                         (72)
     moveto PixMap [1,1];                                             (73)
     JPEGzigzagScan ;                                                 (74)
end                                                                   (75)
```

Bild 21.32: MoPL program of the JPEG scan pattern shown in Bild 22.20.

New directions in supercomputing. Because of high acceleration factors in algorithms, which

Algorithm	6800 16 MHz / millisec	MoM2 10 MHz / millisec	Acceleration factor
CMOS Design Rule Check	91330.20	39.0300	2340
Digital Filter	9126.40	29.4400	310
Lee Routing: seek S	42.50	0.0625	680
wavefront	70.00	0.3750	186
backtracking	23.25	0.1250	186

a)

Algorithm	Data Manipulation	Addressing	Control
CMOS Design Rule Check	7 %	93 %	<1 %
Digital Filter	28 %	58 %	14 %
Lee Routing: seek S	14 %	74 %	12 %
wavefront	6 %	92 %	6 %
backtracking	17 %	67 %	17 %

b)

Bild 21.33: Performance Analysis: a) MoM-2 acceleration factors compared to Motorola 6800, b) overhead analysis on DEC VAX-11/750

are subjects of supercomputing efforts, xputers, their compilers, and their applications are a source of ideas for new directions in supercomputing research - also in compilation techniques because of the paradigm's close relations to data dependency analysis. The xputer execution mechanism supported by scan caches and their address generators is a generalization of vectorization. With xputers the storage schemes for interleaved access memories are derived more easily and can be used for a wider variety of algorithms than with traditional supercomputers. (also see paragraph "Image Processing" in section 22.2). 333

21.11.3 Rapid turn-around ASIC synthesis.

Because the rALU and the use of field-programmable logic the xputer has close relations to ASIC design methods (also see last paragraph in section 22.8). That's why with xputer parts held in cell libraries a new approach to ASIC design could be created. Debugging is by orders of magnitude faster than in traditional ASIC design, since execution is used instead of simulation. By retargeting this programmable version could be converted into a hardwired gate array version for fabrication.

```
ScanPattern LoRzigzagScan is  reverse(rotU(UpRzigzagScan));          (76)
                              :
                              :
                              :
```

Bild 21.35: MoPL program of LoRzigzagScan by transformation of UpRzigzagScan (Bild 22.22).

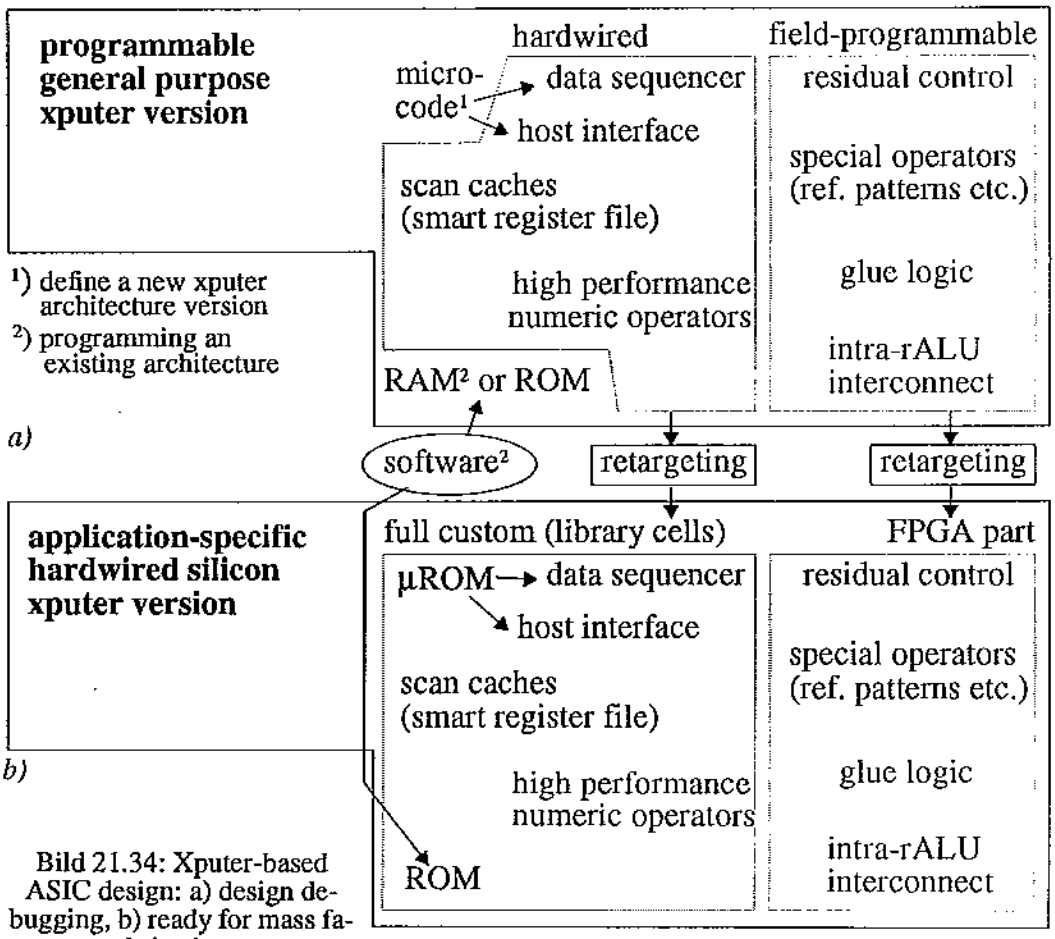

Bild 21.34: Xputer-based ASIC design: a) design debugging, b) ready for mass fabrication.

Bild 22.24 illustrates the xputer-based ASIC design process. Algorithm capture and optimization as well as debugging is carried out on a programmable version of the xputer platform (Bild 22.24a). All "standard circuits" are hardwired, where address generators and external interfaces are microprogrammable (see box named "hardwired in Bild 22.24a), and primary memory is a RAM. By microcode modification the architecture of the target machine may be optimized to a given application area. Machine code generated by the xputer's compiler is loaded into the RAM and into the field-programmable part (see Bild 22.24a).

After debugging an application-specific hardwired silicon version of the machine is derived by a retargeting tool from the hardwired machine version and its machine code (Bild 22.24b). The "standard circuits" (full-custom, if high performance is needed) are fetched from a cell library and their microcode is loaded from machine code of the programmable machine version. The FPGA part (see Bild 22.24b) is derived from the field-programmable part of the programmable

machine version (Bild 22.24a) by retargeting. Let us summarize the advantages of such a design methodology over traditional ASIC design:

- fast turn-around debugging (simulation replaced by debugging),
- designer guidance by a simple machine paradigm as a model behind a high level programming language,
- very high source level: much less complex description input at (e. g. MoPL, or even SYS²: much higher than e. g. VHDL)....
- drastically reduced design effort: the major part of a design consists of general purpose cells from library
- high performance: due to the efficient paradigm high performance results are obtained, although procedural (sequential) methods are used.

Bild 22.23 a gives some performance results having been obtained earlier [58]: acceleration factors, compared to von Neumann implementations of the same algorithms. Bild 22.23b shows some overhead figures obtained experimentally, which partly explain the high acceleration factors.

21.12 CONCLUSIONS

The paper has briefly summarized the new xputer machine paradigm, has demonstrated its basic execution mechanisms, and, has shown its very high efficiency and the reasons for it. The paper has introduced a new high level xputer programming language MoPL-3 being an extension of the language C and has illustrated its comprehensibility and the ease of its use in data-procedural programming for xputers. An earlier version of the language (MoPL-2) has been implemented at Kaiserslautern on VAX station under ULTRIX. For systolizable algorithms a program generator has been implemented as a front end, which generates MoPL programs by using modified versions of projection techniques known from systolic synthesis. The requirements for a novel compilation techniques needed for such data sequencing languages have been summarized. The essential tasks to be carried out by such new compilers and their differences to conventional compilation techniques have been shown by demonstrating the execution of a source algorithm example. It is an essential new aspect of this new computational methodology, that it is the consequence of the impact of field-programmable logic and features from DSP and image processing on basic computational paradigms. 334

It has been shown, that by using multiple scan caches this method can be generalized to cope with all kinds of algorithms with regular data dependencies: with non-locally regular data dependencies, i.e. also for non systolizable algorithms with regular data dependencies. We have illustrated, that xputers, their languages and compilers open up several promising new directions in research and development - academic and industrial. We have shown, that also xputer-based ASIC design is a highly promising new direction of research and development.

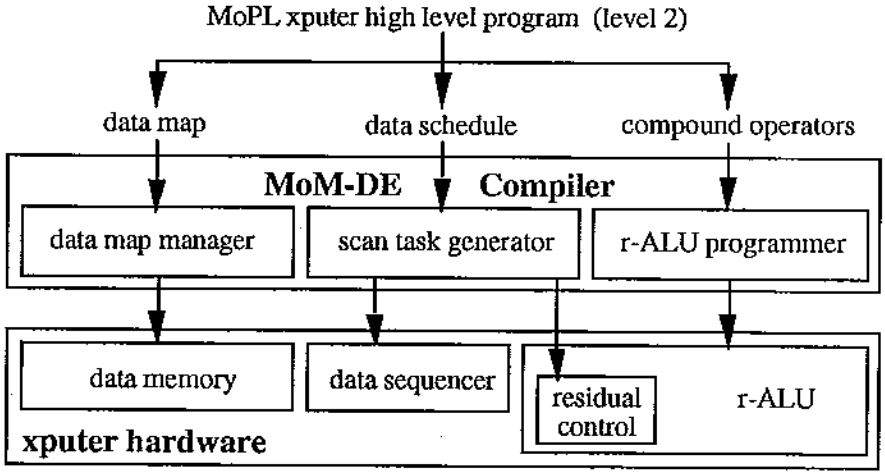

Bild 21.36: Basic Structure of the Compiler of the MoM-DE system