Why Can't Hardware Be More Like Software?

William Sharpe, Dominic McCarthy
Personal Systems Laboratory
HP Laboratories Bristol
HPL-95-16
February, 1995

hardware-software codesign, cosynthesis, information appliance, superchip

This paper recognizes the shift in information appliance development techniques due to the increasing number of transistors that may be accommodated on a piece of silicon. The shift is similar to the one undergone in software as memory sizes enlarged. It is expressed as a willingness of system developers to accept inefficiencies in design for advantages in design time. The tools and complexity issues associated with integrating ever more functionality onto a single die mean that there will need to be a split in the development paths of hardware-software codesign tools, tools aimed at the needs of appliance designers, and tools aimed at the integration of macrofunctions.

1 Information Appliances

The capture, storage and transfer of digital information is growing at a tremendous rate. With world-wide networks, and cheaper access, people will want to be able to interact with this data for commercial and consumer reasons. This implies that consumer demand for appliances that can extract, interact and present that information is going to grow. This trend is forcing computer companies to assess the impact of moving and evolving the technologies, associated with high-powered workstations, into cheap, consumer appliances. Fortunately, integrated circuit technologies are advancing at a rate that will permit such systems to be implemented, *providing* that the tools are developed to enable their speedy design.

Some examples of expected information appliances include:

- multimedia personal computers with high bandwidth communications;
- digital camcorders and cameras;
- set top boxes;
- home imaging printers;
- personal communicators.

It is important to appreciate the characteristics of these appliances: they will exist in the mass market, implying very high volumes; they will have a very low manufacturing cost; they will exhibit low power; they will have a market life of about two years; and the initial product in a family will be developed in less than 18 months. This paper describes the changes in system development technology that are required for this class of appliance to be viable.

2 Accept the inefficiencies!

We believe that over the next few years there will be a major shift in the direction of development of hardware design tools, similar to that which took place many years ago for software. In the early days, all software development was done in assembly code, and programmers had to be very concerned with highly efficient use of all system resources. However, as memory and processor power started to increase rapidly, attention shifted to higher level languages that optimised designer and programmer effort. Similarly, hardware developers are now beginning to ask themselves how to make efficient use of the growing number of transistors that can fit on a piece of silicon, without getting buried by system design complexity. To us, this implies a redefinition of the term efficiency: moving from a measure of the number of gates, to a measure of the functionality, packed onto a piece of silicon.

We believe, therefore, that in future the priorities for system designers of information appliances will no longer be primarily silicon efficiency, but instead will be:

- time to market;
- rapid product turns;

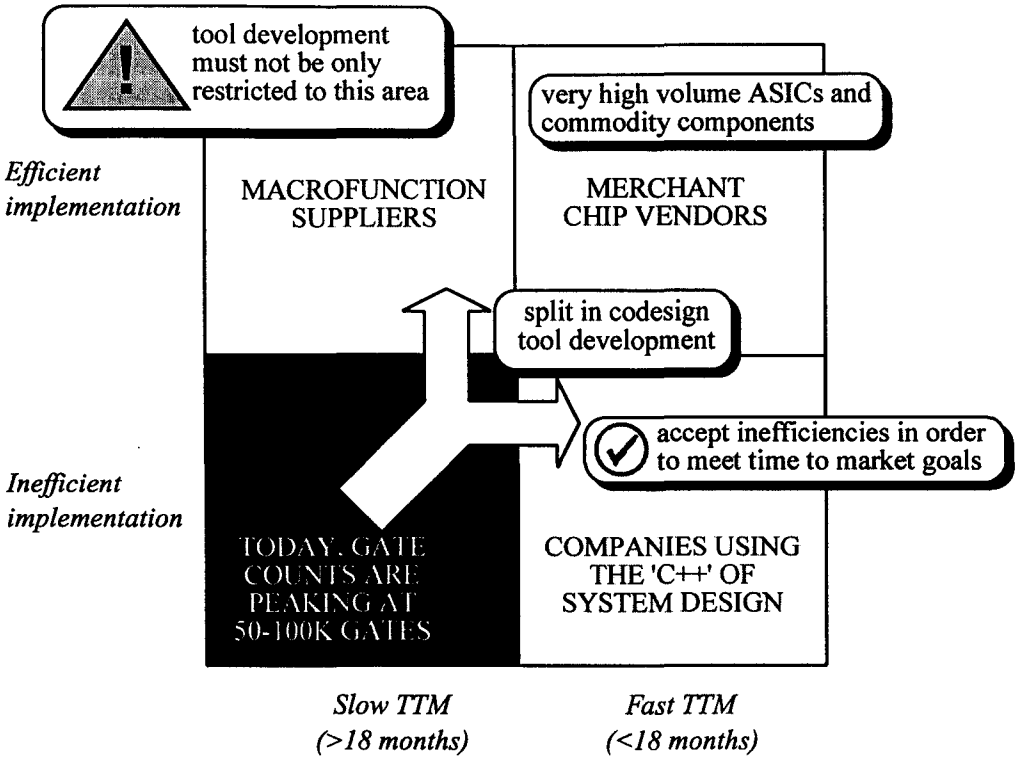


Figure 2.1: Future segmentation from the appliance manufacturer perspective.

- product cost;

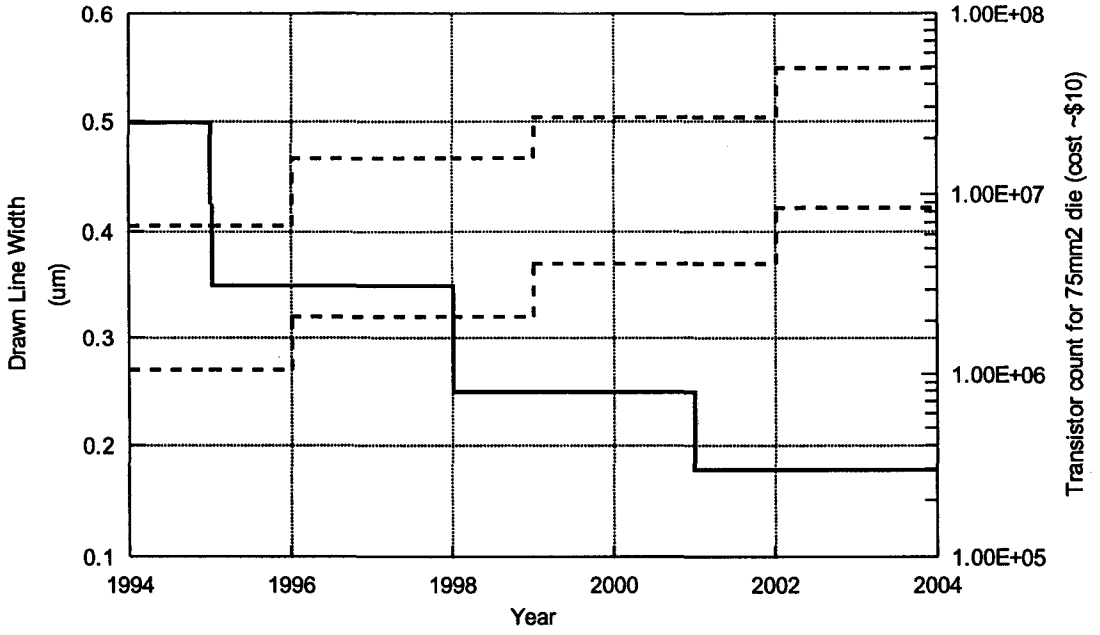
Given this, they will be willing to sacrifice some silicon area in order to meet aggressive design schedules. The cost penalties of a degree of silicon inefficiency will be more than offset by rapid time to market and product turns. We envisage designers using the 'C++' of system design to lay down existing macrofunctions, containing more than the required functionality, in single, 'superchip' solutions. They will accept redundancy and inefficiency provided that design times are reduced. If macrofunctions do not exist, they will tolerate new hardware macrofunction designs of up to around 100,000 gates, and no more, as this figure is proving to be a limit caused by return-on-investment and complexity forces.

There will, therefore, be a split in the development of system design tools to serve the differing needs of appliance designers on the one hand, and specialised merchant chip and macrofunction, or component, providers on the other. Figure 2.1 shows the expected divergence. It is within this framework that we should look at the potential of hardware-software co-design.

3 Technology Trends

3.1 Hardware technology

Integrated circuit (IC) processes are evolving along the curve depicted in figure 3.1. The dashed lines estimate the boundary for a die costing about $10 in volume, using the



Sources: Dataquest Inc. (ASIC-WW-MT-9401) and Hewlett-Packard.

Figure 3.1: Semiconductor technology roadmap.

mainstream IC process (which is shown as occurring 1 year after the process is released). The upper limit roughly defines the maximum number of transistors, assuming an SRAM were implemented; the lower limit is defined using standard cell densities. Today, IC processes offer geometries in the 0.5μm region, with leading edge technologies emerging that offer 0.35μm feature sizes. By the turn of the century, we can expect to have the opportunity to utilise up to 10,000,000 transistors, and up to 50,000,000 a few years later.

Evidence collected from within our company and Dataquest suggests that the majority of IC designs contain between 200-500,000 transistors (or 50-100K gates), and will continue at this level until the turn of the century. Why has this happened? There are four possible reasons:

1. system engineers have reached a comfort level with existing tools;
2. the emergence of new tools, such as behavioural synthesis, do not put orders of magnitude of more gates down than current tools, and any productivity increases are used to improve time-to-market;
3. management of system complexity (both by team and tools) dictates that designs greater than this level are not feasible within the product development time (typically, this is 2 years or less);
4. there are no new tools to improve complexity management - designers can still only achieve about the same in a given time period, since they still need to perform hierarchic decomposition in order to understand their system functionality.

The characteristics of information appliances and their design constraints, force the use of 'superchips'. A superchip is a *single* integrated circuit that comprises several hardware macrocells. An example of a superchip might be a communications processor, constructed from the

macrocell of an existing microprocessor, a multiply-and-accumulate macrocell, some memory and serial port macrocells, or may be a single chip implementation of the Portable Video Phone, described in section 4. 'Macrofunction' is a term that is used to describe the embodiment of some algorithm or functionality in a combined hardware-software component, which is usable by the anticipated tools. Future superchips, developed by system designers, may utilise macrofunctions that are further decomposed into the appropriate hardware macrocell and software objects required.

3.2 Software technology

The evolution of software functionality has matched, if not exceeded, that of hardware. Some of the advances in memory capacity and performance have been sacrificed for improved productivity, sophistication and user 'empowerment'. This development has been characterised by:

- the introduction of extremely high level programming systems, often specialised towards a relatively small problem domain;
- the success of small, entrepreneurial companies, in niche product development;
- object oriented and functional analysis, design and implementation at all levels of sophistication;
- a gradual shift away from the monolithic software structures of the past, towards dynamic systems;
- short version lives for many systems, with rapid and regular upgrading of capabilities;
- mass standardisation around a small number of platforms.

Ten years ago, programmers were a relatively rare breed, supremely familiar with the intricacies of the machines they programmed. At the systems level, there is little difference. What has changed is the population of users who are comfortable manipulating systems that more closely resemble their day to day problems. Mathematicians, control engineers, scientists, etc. benefit from 'programming' tools that allow them to describe a problem, and its solution, in familiar terms. The work of interpreting those descriptions is carried out by powerful, memory-rich computers.

Concurrently, the number of 'intermediate' technology programmers has risen. These are people familiar with languages, such as C++ and Ada, but who need to know little about the underlying services provided by the operating system and hardware.

Numerically (and financially), the software industry has formed an inverted pyramid, with a relatively few number of systems specialists, supporting an enormous industry.

The opening up and standardisation within the software market, may well give rise to a new group of service providers. These will be the companies who provide the small, very

specialised 'objects', which can be bolted together to form complete solutions. Their influence is likely to be felt even by the most naive of users, as standards make it possible for users to customise their applications without recourse to the original developers, and possibly without reference to the developers' original intentions.

4 The portable video phone - a future product?

In order to illustrate our vision of the future, let us consider how system design will be accomplished in the year 2005. For this example, we will consider a product called the Portable Video Phone (PVP). This device will require functionality to support the new cellular standard, GSM-2004, the MPEG-23 codec algorithms, screen rendering and other functionality, such as voice controlled operation.

The system designer will sit at the design station and access the macrofunction library. A block diagram of the functions will be created, illustrating the dataflows between them. The tool will then accept the responsibility for synthesising hardware and software interfaces for the functions, before generating the completed system. This process may be repeated several times, either manually or using heuristics, in order to discover the optimum implementation. Each of the macrofunctions used may have more functionality than required. For example, the MPEG decoder might be able to process all previous 22 formats - this functionality is redundant in the PVP, but, unless it is automatically removed (which might be impossible due to the complexity of the design), then the extra redundancy can be tolerated.

The design of the macrofunction library may also permit some redundancy removal across the chosen macrocells. It is expected that the library components will contain interface definitions similar to those used in software, except that there will be a timing dimension to them. This is obviously an area for further research.

The codesign tools required for this must target the implementation of systems from macrofunction libraries and are responsible for the interface generation. The associated software will be constructed using the same macrofunctions and will be generated from a single system specification. The design of the macrofunction is left to library developers or, provided that the complexity is not too great (i.e. less than about 100K new gates), may be designed in-house using the function synthesis tools. This implies that we foresee a bifurcation in codesign technology: (1) targeted at fast system implementation; and (2) focused on highly efficient macrofunction design.

5 Future scenarios

We have suggested three different segments for the future of system design, as depicted in figure 2.1:

- macrofunction suppliers;

- merchant chip vendors;
- appliance manufacturers.

5.1 Macrofunction Suppliers

The companies operating in this segment will be those who own and evolve specific technologies, and make it available for purchase in the form of macrofunction libraries. These companies will exist because they will focus their efforts on one particular technology and have the skill and resource bandwidth to manage the complexity associated with that technology. They will not have the capability of building complete systems, because that would require the evolution and implementation of too many technologies. Companies fitting this description are emerging today. These deliver high quality, functions which may be directly synthesised, such as processors, and can be easily incorporated into the system development.

The macrofunctions will not be limited to hardware descriptions; they will also include interface definitions and software objects that permit system synthesis, as described in section 4. More advanced libraries may also be able to instantiate multiple architectural options, ranging from pure software, through several different partitions, to pure hardware.

In order to satisfy the tools requirements of these companies, tools will be required to focus on the implementation of algorithms in hardware-software environments. The engineers will want to be able to trade-off architectural options for performance and silicon area. The new behavioural synthesis tools are the first steps in this direction.

5.2 Merchant chip vendors

There will still be a requirement for commodity microprocessors, memory subsystems etc. and these will be generated in high volumes from the major IC suppliers. Here, teams of hundreds of engineers will continue to develop components to satisfy the demand, though the investment will necessitate that the ICs are generic enough that they are adequate for a variety of products, and costing can benefit from economies of scale. These suppliers will attempt to squeeze maximum performance per unit area out of their silicon, to ensure the lowest silicon costs (and, hence, the opportunity for greater profit margins).

It is believed that, as these companies integrate more onto one IC, they will also drive the architectures of the products that utilise them, much in the way that the PC architecture has been driven. This will have effects on the businesses of the system manufacturers who use them. Time-to-market goals may be met by using commodity components, but all competitors will be restricted to the same architecture (and devices) which implies very little cost differential between competitive products. It is also believed that these companies will need to invest in the technology and algorithms used on their ICs. This may mean that key technologies, related to their businesses, will be owned and evolved for proprietary use.

The variety of information appliances, and their need for single 'superchips', produces a different set of requirements than those on which these companies are currently focused. This

may force these companies to either continue to focus on other segments (such as ASICs), or address the issues related to information appliances. Movements in both directions are beginning to be observed.

The system synthesis tools will also be applicable in this segment, though they would use libraries of physical components. Any redundancy could not be removed, and the cost of the product would also be dependent on the number of components used.

5.3 Appliance Manufacturers

This segment contains the system developers who will be utilising the supplied macrofunctions. Their aim is to implement systems extremely quickly. In order to meet this goal, they will tolerate inefficient implementation.

This does not mean that the systems are not implemented well - it means that the emphasis should be on permitting complex systems to be prototyped and partitioned between hardware and software at an architectural level. The tools will utilise macrofunction libraries and interface synthesis, with the final implementation being very fast. The tools have been described in the example in section 4.

6 New industry eco system

Given that the tools are developed, it is interesting to consider the effect on the related industries. We have suggested that a new industry 'eco system' will emerge for the mass market of

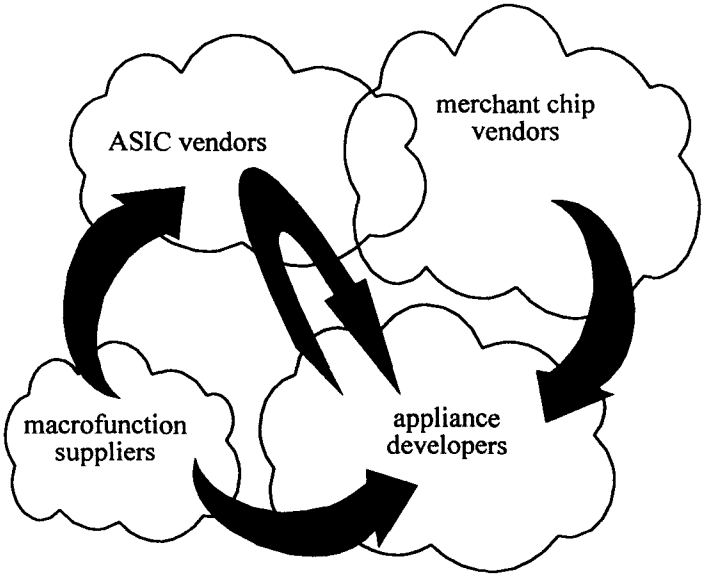


Figure 6.1: Eco system for the mass market of information appliances.

information appliances, that comprises ASIC vendors, merchant chip suppliers, macrofunction developers and system developers, as shown in figure 6.1. The arrows in the diagram indicate supplier-consumer relationships. This chart is intended to provoke discussion of the potential scenarios. Some of the challenges provoked by this chart are:

- will merchant chip vendors split into two core businesses, ASICs and macrofunction suppliers?
- will the macrofunction suppliers grow to become a major player in the industry?
- how much core technology will the appliance developers retain?

7 Conclusion

This paper described how the potential mass market for information appliances will require a new set of hardware-software codesign tools aimed at fast time-to-market and utilising macrofunction libraries. It indicated that tools are required for the macrofunction development, which differ from system tools, and resemble the behavioural compilers of today. Appliance developers would like to see the automation of the system engineering, leaving the engineers to do the system definition, and specific technology evolution, required for our products.

It has also been stated that appliance developers are willing to accept the inefficiencies associated with macrofunctions, in a similar manner to the way that software developers treat memory. The cost of transistors is becoming negligible compared to the cost of managing system complexity, and time-to-market concerns are the major factor in information appliance development. Utilising tools that permit the simple integration of these macrofunctions, much in the same way that complex software systems are developed today, will allow products to be generated successfully.

For the partitioning case in particular, tools will be required for both development paths: partitioning at the system level will involve manipulating macrofunctions; partitioning for macrofunction development will involve the accurate modelling of architectures for algorithm implementation.

Acknowledgements

We would like to thank the many people who have helped contribute to this paper, especially Alan Marshall. Ian Page, Chris Robson, and Nick Wainwright for their useful reviews, and to Herb Gebhart for tracking down much of the integrated circuit information. Most thanks go to Richard Taylor, for his invitation, contributions and support of this paper.