AREA-TIME HIGH LEVEL SYNTHESIS LAWS: THEORY AND PRACTICE

Miodrag Potkonjak
NEC C&C Research Laboratories
4 Independence Way
Princeton, NJ

Jan Rabaey
University of California at Berkeley
Dept. of EECS
Berkeley, CA

Abstract - We introduce three AT DSP high level synthesis laws that relate different components of the area of ASIC implementation cost, namely foreground memory, execution units, and interconnect to the sampling period (available time). The laws state that: $A = const$, $AT = const$, and $AT^2 = const$ for the area of registers, execution units, and interconnect respectively. We validate the AT laws using case studies and statistical analysis of synthesis results of 80 real life designs.

Several applications of the *AT* laws for development of high level synthesis tools are presented. Use of the *AT* high level synthesis laws as an effective method for encapsulation of high level synthesis knowledge is also studied. The effectiveness of the *AT* laws applications is documented on numerous designs.

INTRODUCTION

We study the following design space exploration problem in DSP high level synthesis. An arbitrary ASIC computation is given. When the computation is implemented under a timing constraint that the available time (sampling period) is T_1, the area of implementation is A_1. The question which we are interested in is, if the available time is changed to T_2, what is the area of implementation. Furthermore, we are interested not only in the relationship between the total area of an implementation as a function of the sampling rate, but also in the decomposition of the area requirement over the various design resources: memory elements in data paths (also called foreground memory), functional elements and interconnect. The just outlined problem is a backbone for the crucially important high level synthesis and system level synthesis design exploration task.

Motivational Experiment Efficiency vs. Throughput

The widespread belief is that efficiency and throughput are inversely correlated requirements. The following experiment, however, indicates a surprising evidence: that resource utilization and throughput are strongly positively correlated in high level synthesis ASIC designs.

Table 1 shows the relationship between the AT product and the sampling rate for the 8th order Avenhaus cascade filter [2] obtained using the HYPER [10] high level synthesis system. The data is measured by the size of the datapath at the RT level. AT product is the product of the area of implementation and the sampling period. The following conclusion is apparent: **As the sampling period (available time *T*)**

decreases, the AT product either improves or at worst stays constant.

Sampling period	50	40	30	25	20
EXU (mm^2)	0.52	0.77	0.89	1.09	1.40
EXU AT	26.00	30.8	26.7	27.25	28.00
Registers (mm^2)	2.68	2.68	2.68	2.68	2.68
Interconnects	18	24	36	44	60
INT AT$^{1.35}$	3539	3491	3552	2294	3424
Active Area	3.71	4.17	5.24	5.70	6.17
Active Area AT	26.23	26.37	27.46	26.35	27.59

Table 1: The size parameters of the 8th order IIR Avenhaus cascade filter at the RT level

TIME	50	40	30	25	20
EXU (mm^2)	0.52	0.72	0.89	1.09	1.40
EXU AT	26.00	28.8	26.7	27.25	28.00
REG	27	27	27	27	27

Table 2: The "lower bounds on the size parameters" of the 8th order IIR Avenhaus cascade filter at the RT level

One can argue that "counter-intuitive" experimental results are actually consequence of peculiarities of the high level synthesis tools used. In order to dismiss that objection, Table 2 presents the relationship between min-bounds and the available time. The min-bounds are again provided by the HYPER system, and they indicate lower bounds on the amount of the required resources [10] regardless of the tools used. Again, the experimental data strongly supports the previous conclusion.

TIME	50	40	25
AREA	8.19	8.99	12.65
AT product	57.91	56.86	63.25

Table 3: The size parameters of the 8th order IIR Avenhaus filter at the physical layout level

Furthermore, comparison at the layout level shown in Table 3 indicates again the claimed relationship. The layouts were automatically produced using HYPER and LAGER [3] tools.

In the rest of the paper we demonstrate that the observed relationship in the introductory experiment is not just a fascinating high level synthesis paradox, but a consequence of widely applicable high level synthesis AT laws. The AT laws also present a surprisingly powerful starting point for the development of tools and methodologies, not only in design space exploration and many other high level synthesis tools, but also in a general design process.

Related work

Although CAD and engineering literature describes a number of interesting and important laws (e.g. the famous Brooks [4], Grosch's [6], and Amdahl's [1] laws and Rents rule [5]), the most relevant research efforts are high level estimation techniques and VLSI AT laws in theoretical computer science research.

Jain [7] established the lower-bound area-delay trade-off law which says that the AT product is constant. However, he posed some important assumptions and restrictions in deriving this result: 100% utilization rate of all resources is assumed and only the area of execution units is taken into account. We will show that for the area of execution units this law is valid for a large class of designs, regardless of the resource utilization rate.

The VLSI theory got the initial impetus from the work by C.D. Thompson [14] and was, thereafter, rapidly developed within the theoretical science community. Recently, it has been recognized as an important methodology for parallelism exploration [12]. For excellent treatments on the VLSI AT laws, we refer to [15]. The VLSI AT laws establish bounds on what can be achieved in a given silicon area assuming 100% utilization of all available hardware resources.

HIGH LEVEL SYNTHESIS AT LAWS

This section is devoted to the introduction of three fundamental high level synthesis laws. They are:

1. **Register AT law:** The area of registers in a datapath does not depend on the throughput of the design. So $A = const$ for registers.

2. **Execution Unit AT Law:** The product of the area of execution units and throughput is constant, regardless of a targeted throughput. So $AT = const$ for execution units.

3. **Interconnect AT Law:** The product of the square of the number of interconnects in design and throughput is constant for any throughput. So, $AT^2 = const$ for the number of interconnect.

The following assumptions are instrumental in the derivation of the laws. All designs have in its behavioral description a very large number of operations. Furthermore, for each type of operations present in behavioral description there are many instances.

The available time is significantly larger than the critical path of the computation, but at the same time it is short enough that for all types of operations a large number of execution units is required. There is no structure in the behavioral description of computation and in particular there are no restrictions and regularity in the interconnect pattern of communications (i.e. all communication is global).

Although the presented restrictions appear strict and limiting, the experimental studies show that all restraints can be relaxed, such that the laws can be applied on surprisingly small designs. The only exception is the last constraint (which implies that all communication has to be global),

Experimental Verification

We performed a comprehensive experimental study on 80 DSP, video, image, telecommunication, control and information theory applications examples in order to empirically and statistically validate the high level synthesis AT laws. Particular attention was devoted to ensure a variety of structural characteristics among the considered examples. The diversity of the examples with respect to a number of important design parameters is shown in Table 4.

All examples were studied using the following procedure. Each example was scheduled for at least 5 different available times, starting from the fastest possible (when the available time is set to the critical path time) to the most inexpensive possible implementation (when for each type of resources only one instance of each execution unit type is used). However, in the analysis of results presented in the remainder of this section, only implementations where the available time is at least 4 control steps and at least 10% longer than the critical path are considered, the reason for which will be explained in the next section.

Parameter	min	max	average	median
Number of nodes	20	7,376,121	145,000	159
Number of node types	3	5	4.07	4
CP/ Available Time	0.01	1.0	0.43	0.44
Iteration bound	1	133	7.1	4
Resource Utilization	19%	88%	42%	41%
Area Reg/Exu	0.03	7.14	1.52	1.02

Table 4: Physical parameters of the analyzed examples

By far the most convincing experimental confirmation was obtained for the register AT laws. For 69 examples (86%) the greatest discrepancy in the number of registers at any two different available times was less than 10%. On almost one quarter of examples (19) there was no discrepancy at all. On only 5 examples (6%) the discrepancy was larger than 15%, and the largest discrepancy was only 23%. There were no examples with more than 30 registers which had discrepancy larger than 10%.

The results for the area of the execution units showed larger discrepancy, but still it was apparent that even for small examples the proposed AT law is exceptionally good approximation. For more than half of the examples (45), the largest discrepancy was smaller than 10%. For only 8 examples the largest discrepancy was greater than 25%. The largest discrepancy was 50%. When examples with more than 10 units were considered all examples were within an 11% error margin. For examples with more than 20 units, the largest deviation was less than 5%. Here are two reasons that explain why the register AT law was superior. The first reason is that in all examples, the number of registers was significantly larger than the number of execution units. Recall that the laws were derived under the assumption of a large number of instances of each resource. The second reason is that in several examples the area of the execution units is dominated by the area of the big array multiplier, and a change in the number of multipliers between a narrow range of available times results in big difference in the corresponding AT product.

The interconnect AT^2 law results from the assumption that the interconnect pattern is *fully random and fully global.* We did not find a single example for which this law was completely correct. However, for all examples the number of interconnects was between the AT and AT^2. It is interesting to note that actually almost all examples were following curves dictated by functions of the type AT^r, where r is varying between 1.2 and 1.5. Nevertheless, we decided to preserve the original AT^2 interconnect law in our presentation for the sake of simplicity of intuitive reasoning and because it is implied by the set of assumptions. Obviously, when this law is used the just mentioned observations have to be taken into account.

APPLICATIONS OF HIGH LEVEL SYNTHESIS AT LAWS FOR DESIGN SPACE EXPLORATION AND IN ASIC DESIGN PROCESS

This section presents a brief description of the wide spectrum of applications, for which the AT high level synthesis laws are of interest. They can be divided in two groups. The first group covers the application of the high level synthesis AT laws for CAD tool development. The second group includes applications where the AT laws are used for efficient encapsulation of design knowledge, which consequently can be used either to guide design process or to reason about fundamental qualities of high level synthesis approaches and algorithm.

Using High Level Synthesis AT laws for Estimations and Development of High Level Synthesis Algorithms

The applications of the AT laws itself can be divided into two categories. The laws can be used to derive time efficient, accurate and consistent prediction tools. This tools can be used as an objective function when conducting optimization on a higher level of abstraction. The use of AT laws for the development of high level synthesis techniques (such as module selection, clock cycle selection for power optimization, and hierarchical allocation and scheduling) is described in a related technical report [9].

It is currently being recognized that fast and accurate prediction (estimation) tools are an integral component of high level synthesis [7], [10]. The AT high level synthesis laws provide a base for development of prediction tools with a variety of accuracy/speed trade-off's.

For example, if the highest priority is rapid prediction, we can use the following two step procedure for the estimation of the area of execution units for some arbitrary available time *T*.

(1) *Allocate for each type of execution unit resource only one instance.*

(2) *Schedule the given CDFG in the shortest possible time under the outlined resource allocation resources.* Note that the scheduling task can be done very rapidly in this case (in contrast to general scheduling and assignment), as the whole assignment is fixed and known.

(3) *Using that AT = const for execution units, calculate the area requirements for a given T.*

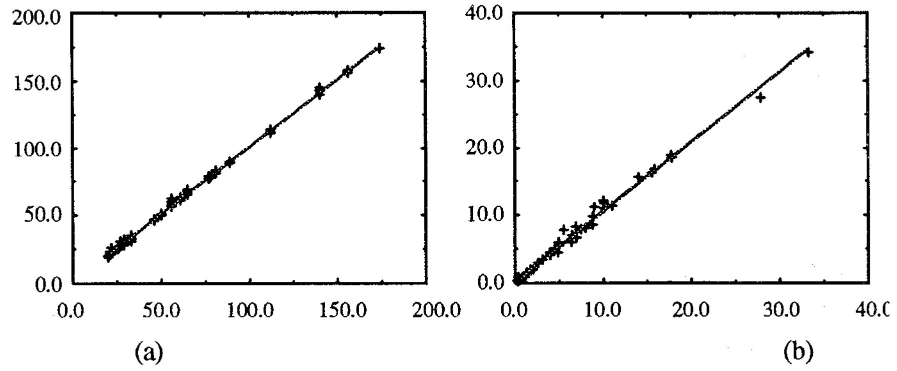

Figure 1: (a) Building prediction models using the AT laws theory: Predicting the number of registers; (b) Building prediction models using the AT laws theory: Predicting the area of execution units

A similar procedure can be used for the number of interconnect (which can be done only in one way when only one instance of each type of execution units is available) and registers. However, note that in the case of interconnect, it is mandatory to obtain the measure of globality of transfers in the corresponding CDFG. Figures 1a and 2b show the prediction models using linear regression.The correlation analysis shows that the correlation between the prediction variable and data obtained after scheduling is 99% for both registers and execution units models build using the AT laws.

The AT laws prediction tool can be directly used in many high level synthesis tools to replace currently used objective functions. For example, it can be used for algorithm and architecture selection, partitioning, transformations, and allocation, assignment, and scheduling. For details of all mentioned applications of estimations tools see [10], [11].

High Level Synthesis Knowledge Encoding

As high level synthesis matures, a growing need for development of a sound theoretical foundation arises. The AT high level synthesis laws are an effective way to encapsulate a part of the high level synthesis knowledge. We now demonstrate how AT laws can be used to encode synthesis knowledge and therefore guide the design process. A more comprehensive description of several other methods and applications of the high level synthesis AT laws is given in [9].

Throughput-Efficiency Relationship. We start by identifying criteria for positive correlation between throughput and efficiency. There are at least two situations when the AT product can be significantly improved, while simultaneously increasing the throughput of the implementation:

1. When the cost of design is dominated by the cost of registers. This situation is common when a high level of hardware sharing is used, or when the cost of registers is relatively high to the cost of execution units and interconnect (e.g. a fixed point computation without multiplication, which uses relatively few bits in its data representation).
2. The second situation can be spotted by analyzing if an assumption used in the derivation of the AT laws is not fulfilled. For example, when some type of nodes (in particular with high implementation cost, e.g. multiplications) have very few instances in the considered CDFG. For example, if we have only 2 multiplications in the CDFG and use 10 control steps, the design is bound to be inefficient. Changing the available time to very few control steps will improve resource utilization.

Example	Area			Area x Time		
	T = 20	T = 40	T = 60	T=20	T=40	T=60
FIR Hamming	11.26	6.84	6.17	225	274	370
DS FIR	8.90	7.55	6.50	178	302	390
iir6	4.44	3.26	2.00	89	130	120
gm	2.03	1.68	1.18	41	67	71

Table 5: Improving the AT products due to constant register requirements. T denotes the available time. The area of registers for all examples were uniform regardless of the sampling rate. The area of registers were 5.22, 5.17, 1.62, and 0.96 for FIR Hamming, DS FIR, iir6, and gm respectively.

Table 5 shows 4 examples which clearly confirm the first claim at the RT level. The reported area is the active area which includes the area of execution units, registers,

TIME	1	2	3	4	6	10	15
AREA	73.58	84.90	75.90	73.83	62.40	23.83	7.44
AT	74	170	228	295	374.40	238	112

Table 6: Improving the AT products - Physical layout data - COLOR transform example

and multiplexers. Table 6 reports data at the physical layout level for a COLOR transform example. The AT product has improved more than twice as time is reduced from 6 control steps to 2 control steps. The implementation with no hardware sharing (when the available time is 1 control step) is even smaller, but here the analysis indicates that the improvement is coming mainly from the absence of the complex hardware sharing control logic. The AT product of the two slowest implementations is reduced due to drastically reduced interconnect. The typical example which supports the second claim is the 2nd order pipelined Volterra filter where we see only slightly more than 10% change in the area as the throughput is doubled as shown in Table 7. This is due to the fact that for all implementations only one multiplier is sufficient and dominates the total area.

TIME	EXU AREA	AREA	AT product
15	13.44	23.42	351
20	13.19	22.65	453
30	12.69	21.22	637

Table 7: The active area of the 2nd order pipelined Volterra filter

Interconnect importance. Next, the AT laws reaffirm well known observations that the importance of local interconnect cannot be overstated in high performance designs. Since interconnects grow as the second order polynomial with the throughput improvement, while register requirements are constant and execution unit cost grows only linearly with increased throughput, it is apparent that interconnect cost will eventually dominate cost of high performance designs. The results from Table 8 strongly support this conclusion.

Example	T = 20		T = 40		T = 60	
	Mux	Tot	Mux	Tot	Mux	Tot
FIR32	3.02	7.41	0.85	4.73	0.52	4.47
Conv5	1.54	3.45	0.37	1.51	0.22	1.36
ellip	3.13	6.69	0.84	3.36	0.35	2.15
126FIR	10.85	14.27	2.61	6.68	0.62	3.38

Table 8: Importance of optimizing Interconnect at high speed. T is the available time. The area of multiplexers (Mux) is rapidly becoming the dominant part of the overall implementation area (Tot)even when only associated control logic is considered

Researchers and developers can also use the developed AT laws in many other ways. Due to space limitation, we will mention only one of them.

Benchmark and Driver Example Selection. There exists one more very important and, at first. paradoxical consequence of the AT laws: *When a high resource utilization is targeted, implementation in close or equal to the critical path length should be avoided.* For this class of implementations, a much higher additional cost has often to be paid for very small increase in performance.

Scanning the high level synthesis literature it is evident that almost all published benchmark examples are done so that available time is either set to the critical path or just slightly longer. For example, a number of high level synthesis systems require 3 adders and 3 multipliers when implementing the 5th order elliptical wave digital filter in 17 control steps. This is proven to be the minimum hardware configuration. When 21 control steps are available, only 2 adders and 1 multiplier are sufficient. So, we see that for a 19% increase in throughput, almost a 3 times higher price (a multiplier is significantly more expensive than the adder in fixed point) in the number of execution units has to be paid. The difference in required interconnect is even more dramatic. Analysis of the results reported in Tables 9 and 10 once again convincingly demonstrates the correctness of the consequence of the AT laws

Name	CP	CP Area	Time T1	Area T1	CP/T1	CP Area/ Area T1
iir11Ch	28	88.67	40	31.83	0.70	2.79
wave5	15	8.57	20	4.50	0.75	1.90
IIR-df	18	67.50	20	23.40	0.90	2.88

Table 9: Exceptionally high cost when scheduling in the critical path time. The reported area is the active area

Name	CP	CP Area	TimeT1	AreaT1	CP/T1	CPArea/ AreaT1
cascade	10	67.54	25	8.19	0.40	8.28
GM	21	13.22	30	5.15	0.70	2.57
CF	28	36.94	30	18.41	0.93	2.01

Table 10: Exceptionally high cost when scheduling in the critical path time. The reported area represents the physical layout data (in mm^2).

CONCLUSION

We established the following three fundamental AT laws in high level synthesis: $A = const$ for registers, $AT = const$ for execution units, and $AT^2 = const$ for interconnect. The laws are experimentally verified and used as a basis for the development of a number of high level and system synthesis tools.

REFERENCES

[1] G.M. Amdahl, et al. : "Architecture of the IBM System/360", IBM Journal of Research and Development, Vol. 8, No. 2, pp. 87-101, 1964.

[2] E. Avenhaus: "On the design of digital filters with coefficients of limited word length", IEEE Trans. on Audio and Electroacoustics, Vol. 20, pp. 206-212, 1972.

[3] R.W. Brodersen, ed.: "Anatomy of a Silicon Compiler", Kluwer Academic Publishers, Boston, MA, 1992.

[4] F.P. Brooks: "The mythical man-month: essays on software engineering", Addison-Wesley, Reading, MA, 1975.

[5] W.E. Donath: "Wire length distribution for placement of computer logic", IBM Journal of Research and Development, Vol. 25, No. 3, pp. 152-155, 1981.

[6] H.R.J. Grosch: "High Speed Arithmetic: The Digital Computer as a Research Tool", Journal of the Optical Society of America, Vol. 43, No. 3, pp. 306-310, 1953.

[7] R. Jain: "High-Level Area-Delay Prediction with Application to Behavioral Synthesis", Technical Report CENG 89-23, Electrical Engineering- Systems Department, University of Southern California, Los Angeles, CA.

[8] M.C. McFarland, A.C. Parker, R. Camposano: "The High-Level Synthesis of Digital Systems", Proc. of the IEEE, Vol. 78, No. 2, pp. 301-308, 1990.

[9] M. Potkonjak, J. Rabaey : "Area-Time Laws in High Level Synthesis: Theory, Validation, and Applications", Technical Report, NEC USA, 1994.

[10] J.M. Rabaey, at al. "Fast Prototyping of Datapath-Intensive Architectures", IEEE Design and Test, pp. 40-51, 1991.

[11] J.M. Rabaey, M. Potkonjak: "Estimating Implementation Bounds for Real Time DSP Application Specific Circuits", IEEE Trans. on CAD, Vol. 13, No. 6, to be published, 1994.

[12] C.L. Seitz: "Concurrent Architecture", in "VLSI and Parallel Computation", ed. by R. Suaya, G. Birtwistle, Morgan Kaufmann, San Mateo, CA, 1990.

[13] R.A. Thisted: "Elements of Statistical Computing", Chapman, New York, NY, 1988.

[14] C.D. Thompson: "Area-time complexity for VLSI", Symposium on the Theory of Computing", pp. 81-88, 1979.

[15] J.D. Ullman: "Computational Aspects of VLSI", Computer Science Press, Rockville, MD, 1984.