A New Knowledge-Based Design Manager Assistant for CAD Frameworks

F. Moreno and J. Meneses

Technical University of Madrid
E.T.S.I. Telecomunicación
Ciudad Universitaria s/n
(28040-Madrid) Spain

Abstract

In this paper we introduce a new knowledge-based method for planning and managing the VLSI design process, based on prediction and advice, that minimizes search in a wide design space. It draws on a newly developed system able "to learn" to collect detailed information on the physical, technological and logical properties of the primitives available for building the design. This may help reduce time from market and may make it much easier for people to learn the tool and issues related to conceptual design. A prototype knowledge-based design manager, called ViPtool, that uses this method is presented.

1 Introduction

The necessity of new design methodologies in VLSI design is nowadays a well-known problem. Competitiveness in the VLSI design area imposes some premises that must guarantee good design quality. On the other hand, in order to achieve maximum success in the development of a VLSI design, great attention must be placed at the architectural design level.

As the number and diversity of computer-aided design tools used by VLSI circuit designers continues to grow, the need for CAD frameworks increases. In general, CAD frameworks provide such services as design data management and CAD tool encapsulation.

Some frameworks provide capabilities at a very high level, allowing the user to deal with the problem from the point of view that the design is a set of functions independent of the tool used [1].

Recently, some tools have arisen with very high abstraction levels. They allow graphical or texual problem specifications based upon a set of functional constraints [2]. The range and complexity of those problems are very poor. Even tools such as Clio [3] (*Conceptual Design framework*) have arisen. They provide the designer assistance at the quantitative level, which comprises the prediction of parameters such as "area", "power consumption", etc.; and at the qualitative level or design advice level which relates to a specific problem. The final target is to reduce the search space of possible solutions. However, there is already, in all of the examples mentioned above, a very high lack of flexibility, because all of them are problem-oriented, technology-oriented, and above all, the "dialogue" framework-designer is limited by a set of static data bases.

In order to support a decision during conceptual design together with high design flexibility, it is necessary to provide frameworks with a *Conceptual Design Framework* able to adapt itself to different design situations, technology, etc., to establish a run-time "dialogue" between the tool and the designer. This will increase the framework performance, and such a performance has been achieved by **ViPtool** by means of a *Knowledge Based System* (KBS).

ViPtool is, at the present, integrated with VANTAGE (Vantage Analysis Sys., Inc. 1987-1992) and is currently being integrated with SYNOPSYS (Syn., Inc. 1988-1992).

2 Previous Related Research

Before beginning the discussion of ViPtool, it is useful to compare it to Clio [3] and Yoda [4]. Some of the principles of design managed in Yoda were relatively general, however their actual implementation was specifically focused on the conceptual design of DSP filters. Yoda contained a "domain specific" assistance subsystem that provided advice and performance

predictions to aid the designer with decision making during conceptual design. Furthermore, Yoda was not meant to be a general design assistance facility to model the general hierarchical and modular nature of design, nor was it meant to generate a general taxonomy of prediction models for the entire VLSI domain. In addition, Yoda "hardcoded" a specific hierarchical structure for the prediction models it contained.

On the other hand, Clio operated as the collective memory of the entire community of designers utilizing the framework, providing for the interchange of information and experience. So, all advice was tagged with information that aided the designer to determine design validity. In order to be able to get quantitative prediction of performance characteristics, such as "area", "power dissipation", etc. from a set of specifications, the activities of synthesizing and fabricating designs had been modeled.

Clio used a knowledge base organization to store prediction models. Whenever a prediction was requested in the context of a given design problem, Clio searched for the prediction model to be used. If such a prediction model was absent, Clio moved up in the hierarchy, until it found a template containing a prediction model that evaluated the figure of merit specified ("area", "power", etc.) by the prediction request. This procedure will be more accurate as the number of predictors grows. Clio was not able to accept new predictors at run-time, so it was necessary to rebuild the knowledge base and to start the design process over again.

A new level of flexibility is needed, in order to really offer a general purpose design framework able "to learn" at run-time, as is the case of ViPtool. Most of the tools reported are focused on particular problems and the specifics of these problems are "hardcoded" [5][6]. This type of systems may not lead to better design from expert designers, but it may help reduce time from market and may make it much easier for people to learn the tool and issues related to conceptual design. This is a big issue in the EDA industry.

On the other hand, ViPtool is able to build a VHDL specification by means of a set of generic and parametrizable libraries. The user may then do a functional simulation in order to achieve a "fast" prototype. The methodology that implements ViPtool is, therefore, top-down because the VHDL libraries are sinthesys oriented and bottom-up, because it evaluates technological parameters such as "area", "number of gates", etc at a very early stage of the design (to be explained in next sections), all together with a fast prototyping capability.

3 Defining the Knowledge-Based Design Manager Problem

Conceptual design is the process of decision making that is required before a detailed design is undertaken. A conceptual design is completed when some *design issues* have been identified by the designer, for instance: "fabrication technology" (1.0, 1.2, 1.5 microns, etc.) and the *design options* such as "core limited", "high core limited", "package type", "pin-out", etc. The knowledge-based design manager is used during conceptual design to aid the designer in making decisions by giving guidance on the most promising design alternatives. A "dialogue" is set up between the knowledge-based design manager and the designer. Thus, ViPtool guarantees very early and accurate prediction models, facilitating the definition and introduction of new prediction models at run-time.

In general, as is mentioned above, there are two different categories of assistance, that can be provided and/or modified during conceptual design: qualitative design advice (consisting of pieces of information about technology and physical data) and prediction (in order to obtain quantitative prediction of performance characteristics, such as "area", "power dissipation", "frequency", "number of gates", etc.).

4 ViPtool: A Prototype Design Manager

A typical prediction process starts with the definition of the target design domain. ViPtool has been tested in the domain of VLSI architectures for speech recognition. Speech recognition for vocabularies larger than 100 words requires, usually, a very high computational load. For speaker-independent speech recognition (isolated utterances) with vocabularies up to 1000 words, it is necessary to use up to 1000MIPS [7]. The Viterbi algorithm is one of the most frequently used in speech recognition. It uses Hidden Markov Models (HMM) to represent phonemes (in a phoneme model) or words (in a word model) [8]. Due to the specific nature of these VLSI architectures, it has been necessary to build a particular software module that implements heuristics depending on the speech recognition algorithm, in order to obtain accurate information about architectural parameters (memory, buses, etc.).

ViPtool is based upon a KBS able "to learn" [9][10][11]. The KBS retrieves information from a table, an Examples Table (ET) built by the designer. This table is formed by a set of attributes also defined by the user (See Figure 1, next page). The designer introduces values for each attribute and matches them with a generic

Figure 1. Examples Table

architecture. This information must be complete and consistent (without uncertainty) and can be retrieved during run-time to be updated and/or modified.

The KBS, then, will modify its own behavior, building a *decision tree* (ID3 Algorithm [10][11]).

Each node tree is an attribute and each branch between nodes is an attribute value. The decision tree finishes with the generic architectures ("tree leaves"). The tree is built calculating the maximal enthropy reduction [10][11].

The ID3 Algorithm is based upon the basic algorithm CLS (Concept Learning System)[10] and works over the whole ET. At the beginning, the KBS has a maximal enthropy:

$$H(N0) = -\sum_{i=0}^{i=k} (P_i * \log_2 P_i) \qquad (1)$$

Where P_i is:

$$P_i = \frac{n_i}{n_1 + n_2 + \ldots + n_k} \qquad (2)$$

$$0 \le H(N0) \le \log_2(Nº Class.) \qquad (3)$$

N0: root
And k the number of different architectures defined in the ET.

Then, the probabilities may be calculated for each generic architecture and $H(N_0)$ as follows:

$$H(N0) \le \log_2 6 = 2.58 \Rightarrow E.Max \qquad (4)$$

$$P_{C_1} = \ldots = P_{C_6} = \frac{1}{9}\,;\, P_{C_3} = \frac{2}{9}\,;\, P_{C_5} = \frac{3}{9} \qquad (5)$$

$$H(N0) = 2.41937 \qquad (6)$$

So, which is the "best" attribute?. Or, in other words,

Figure 2. ViPtool Decision Tree

which is the attribute that gives the most information to the KBS?. The KBS will take into account the first attribute (Nº STATES). There are (see Figure 1) 8 examples with NºSTATES=3 and 1 with NºSTATES=5 (there are not any more values for this attribute), then:

$$P(N^{\underline{o}}ST=3)=\frac{8}{9};P(5)=\frac{1}{9} \quad \textbf{(7)}$$

$$H_{N^{\underline{o}}STATES=3}=2.5 \quad \textbf{(8)}$$

$$H_{N^{\underline{o}}STATES=5}=-1*\log_2 1=0 \quad \textbf{(9)}$$

$$H[NO,N^{\underline{o}}ST]=2.22 \quad \textbf{(10)}$$

Then, the information gain is:

$$G=H(NO)-H[NO,N^{\underline{o}}ST]=0.197 \quad \textbf{(11)}$$

This procedure will be performed by the KBS recursively for all the attributes, in order to get the "best" one (Maximal Gain), and all the examples belonging to the ET will be tested. If some of them are not yet well classified (without uncertainty) then, the procedure, will be repeated again (recursively) until that. As a result of that, a Decision Tree is built (See Figure 2). At run-time, the designer, can add (delete or modificate) the ET and the KBS will make a new Decision Tree.

This KBS has been tested up to 200 examples tables and up to 20 attributes per example, yielding a very good performance. For bigger tables it is necessary to work with subsets of the ET (working set)[8].

The CLS algorithm may be summarized as follows:

procedure CLS (ET)
  if all the examples belong to C_j **then**
    finish C_j;
  else
    new node=the "best" attribute;
    make subtables for each attribute value;
  foreach subtable
    CLS(subtable);
  end;

Design assistance begins with the designer specifying some attributes values. The KBS searches through the decision tree until a generic architecture is reached (See Figure 2). Then, the designer must introduce a collection of discrimination factors (i.e., fabrication technology) and/or design options ("core limited", "pad limited", etc.) (See Figure 3).

Figure 3. ViPtool Discrimination Factors

The designer is free to select some design issues, so, if some problem and/or inconsistency arises, ViPtool informs the designer about possible solutions maintaining a run-time "dialogue". After this job is successfully accomplished, the designer is informed about prediction information (quantitative), such as, "core area", "number of gates", "package suggested", etc.) (See Figure 4).

Figure 4. ViPtool Quantitative Information

ViPtool is, at the present, integrated with VANTAGE. Therefore, if everything is right, the designer can select the VANTAGE interface window. ViPtool sends VANTAGE all the information necessary to parametrize VHDL libraries and to simulate the final specification in VHDL, showing the results in a VANTAGE window.

5 Conclusions

In this paper we have discussed and extended a general design assistance facility and the basic methodology that should underlay its implementation. ViPtool, a knowledge-based design manager based on prediction and advice that has successfully achieved our proposed methodology, was described. ViPtool is currently integrated with VANTAGE and in the near future, will also be fully integrated with SYNOPSYS.

Acknowledgments

The authors are grateful to S. Keller for suggesting improvements to this paper.

6 Bibliography

[1] J.B. Brokman and S.W. Director, "The Hercules CAD Task Management System". In *Proceedings of the IEEE International Conference on Computer-Aided Design, IEEE*, 1991.

[2] M.F. Jacome and S.W. Director, "Design Process Management for CAD Frameworks". In *Proceedings of the 29th ACM/IEEE Design Automation Conference*, IEEE Press, 1992.

[3] J.C. López, M.F. Jacome and S.W. Director, "Design Assistance for CAD Frameworks". In *Proceedings of the EURO-DAC'92*, IEEE Press, 1992.

[4] A.M. Dewey and S.W. Director, "Yoda: A Framework for the Conceptual Design VLSI Systems". In *Proceedings of the IEEE International Conference on Computer-Aided Design*, IEEE, 1989.

[5] R.D. Müller-Glaser, K. Kirsch and K. Neusinger, "Estimating Essential Design Characteristics to Support Project Planning for ASIC Design Management". In *Proceedings of the IEEE International Conference on Computer-Aided Design*, IEEE, 1991.

[6] F.J. Kurdahi and A.C. Parker, "Techniques for Area Estimation of VLSI Circuits". *IEEE Transactions on Computer-Aided Design, 8(1)*, January 1989.

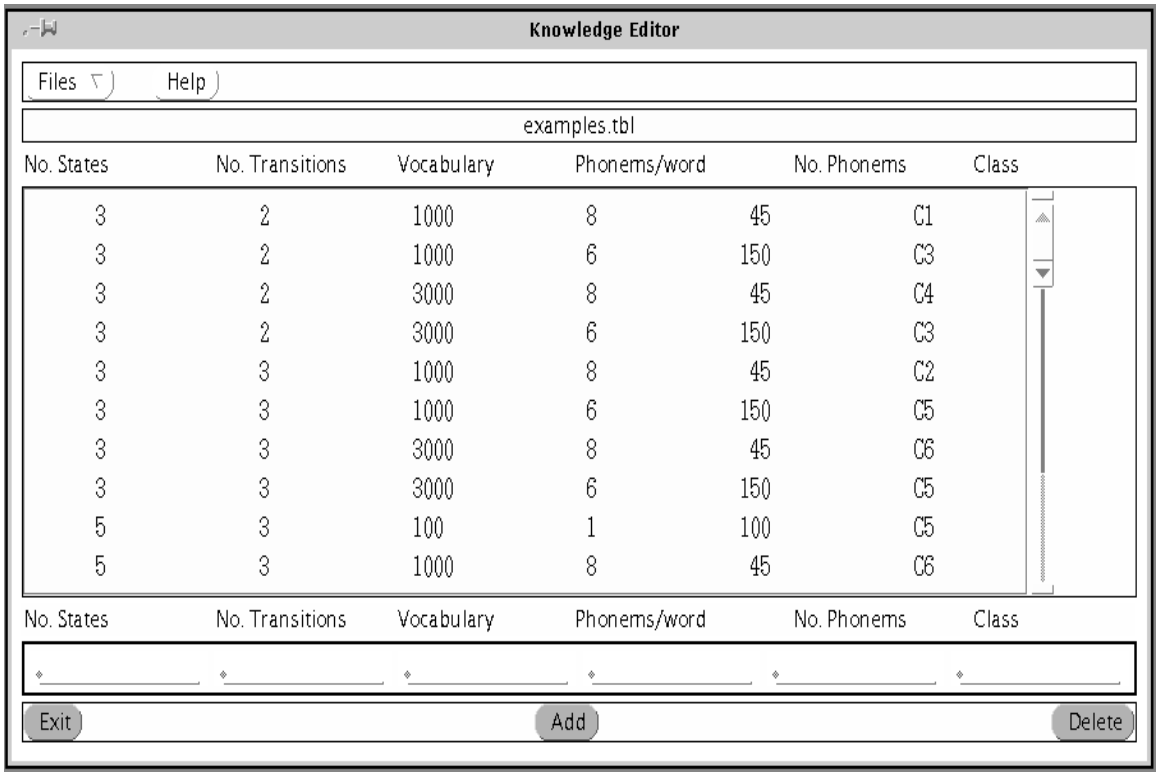
[7] L. Rabiner, "Speech Recognition". *DSP Review, Vol. 2, No. 3*, ATT, October 1989.

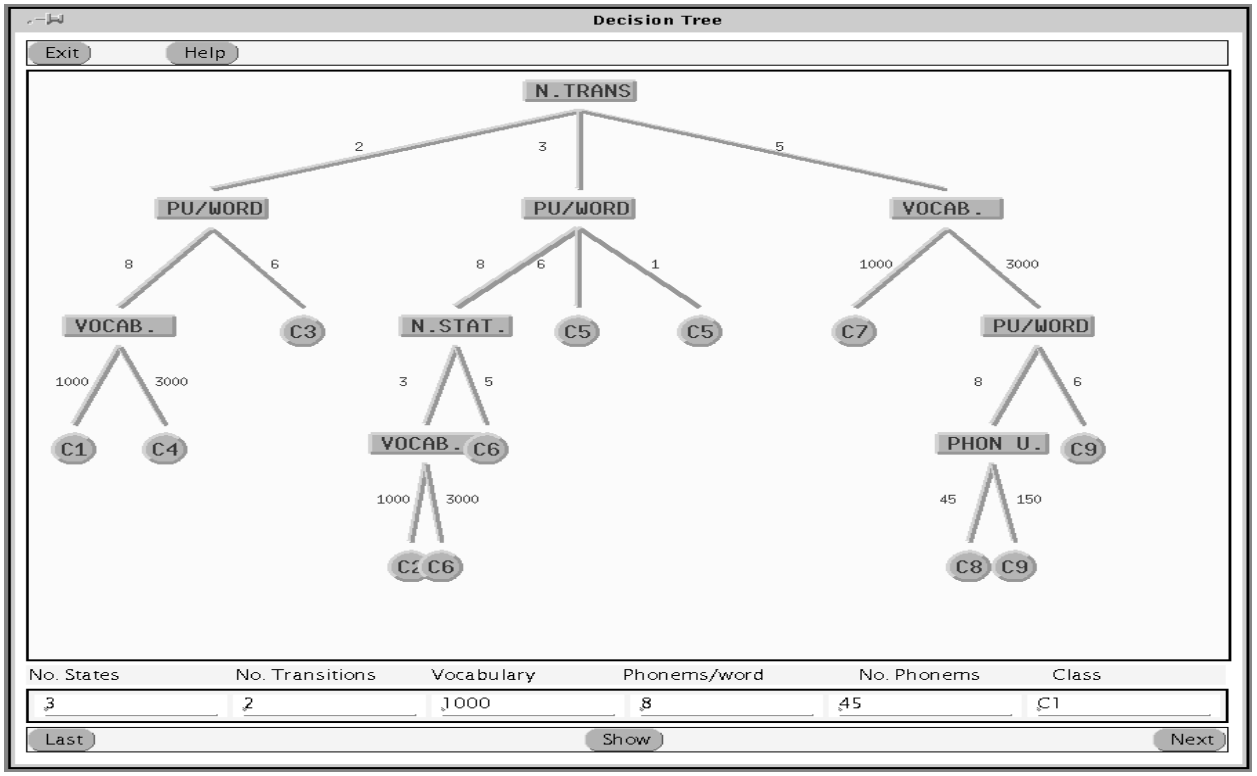
[8] S.J. Cox, "Hidden Markov Models for Automatic Speech Recognition: Theory and Application". *British Telecom. Technology Journal, Vol. 6, No. 2*, April 1988, pp. 105-114.

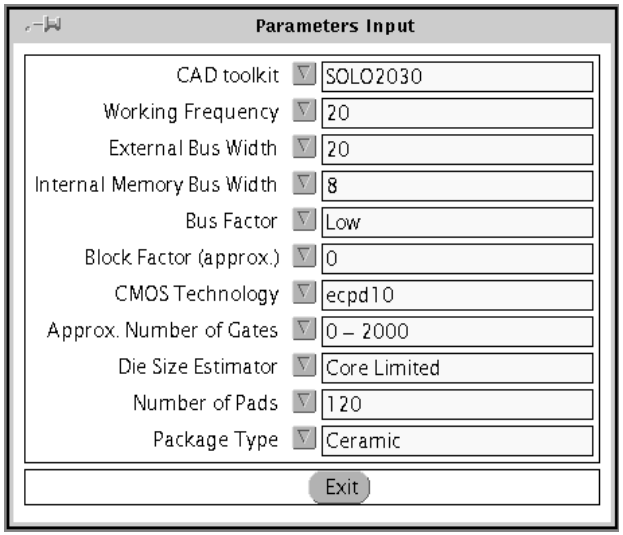
[9] R.S. Michalski et. al., "Machine Learning: An Artificial Intelligence". *Vols. 1-2, Eds. Morgan Kauffmann, 1983-86, Springer-Verlag*.

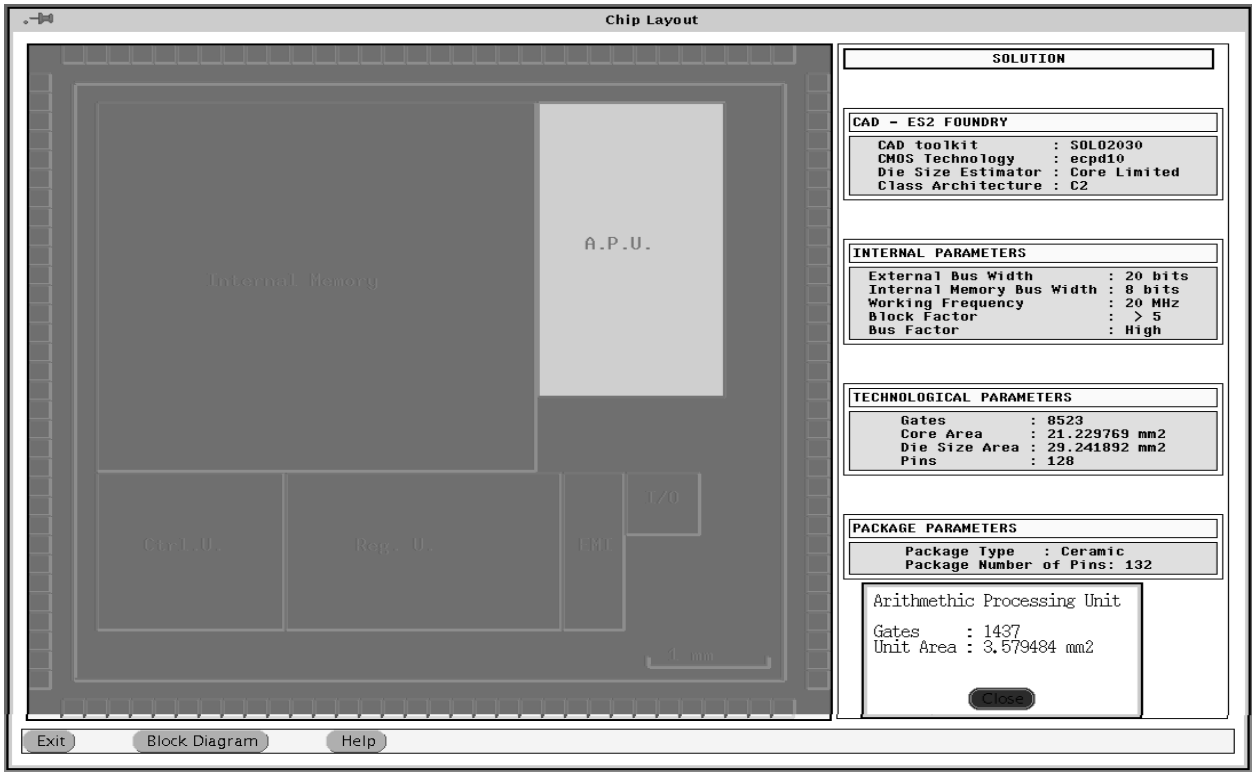
[10] J.R. Quinlan et. al., "Interactive Dichotomizer, ID3". *Eds. Morgan Kauffmann, Springer-Verlag*, 1979.

[11] J.R. Quinlan, "Introduction of Decision Trees". In *Readings in Machine Learning. Eds. Morgan Kauffmann*, San Mateo, Calif., 1990, pp. 57-69.


Knowledge Editor
Files
Help
examples.tbl
No. States
No. Transitions
Vocabulary
Phonems/word
No. Phonems
Class
3 2 1000 8 45 C1
3 2 1000 6 150 C3
3 2 3000 8 45 C4
3 2 3000 6 150 C3
3 3 1000 8 45 C2
3 3 1000 6 150 C5
3 3 3000 8 45 C6
3 3 3000 6 150 C5
5 3 100 1 100 C5
5 3 1000 8 45 C6
No. States
No. Transitions
Vocabulary
Phonems/word
No. Phonems
Class
Exit
Add
Delete


Decision Tree
Exit
Help
N.TRANS
2
3
5
PU/WORD
PU/WORD
VOCAB.
8
6
8
6
1
1000
3000
VOCAB.
C3
N.STAT.
C5
C5
C7
PU/WORD
1000
3000
3
5
8
6
C1
C4
VOCAB.
C6
PHON U.
C9
1000
3000
45
150
C2
C6
C8
C9
No. States
No. Transitions
Vocabulary
Phonems/word
No. Phonems
Class
3
2
1000
8
45
C1
Last
Show
Next


Parameters Input
CAD toolkit
SOLO2030
Working Frequency
20
External Bus Width
20
Internal Memory Bus Width
8
Bus Factor
Low
Block Factor (approx.)
0
CMOS Technology
ecpd10
Approx. Number of Gates
0 – 2000
Die Size Estimator
Core Limited
Number of Pads
120
Package Type
Ceramic
Exit


Chip Layout
A.P.U.
Internal Memory
Ctrl U.
Reg. U.
FMI
I/O
1 mm
SOLUTION
CAD - ES2 FOUNDRY
CAD toolkit : SOL02030
CMOS Technology : ecpd10
Die Size Estimator : Core Limited
Class Architecture : C2
INTERNAL PARAMETERS
External Bus Width : 20 bits
Internal Memory Bus Width : 8 bits
Working Frequency : 20 MHz
Block Factor : > 5
Bus Factor : High
TECHNOLOGICAL PARAMETERS
Gates : 8523
Core Area : 21.229769 mm2
Die Size Area : 29.241892 mm2
Pins : 128
PACKAGE PARAMETERS
Package Type : Ceramic
Package Number of Pins: 132
Arithmethic Processing Unit
Gates : 1437
Unit Area : 3.579484 mm2
Close
Exit
Block Diagram
Help