A Methodology and Design Tools To Support System-Level VLSI Design

Kayhan Kucukcakar and Alice C. Parker

CENG Technical Report 94-18

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-4476

June 1994

A Methodology and Design Tools To Support System-Level VLSI Design

Kayhan Küçükçakar and Alice C. Parker

June 19, 1994

Abstract— System-level design involves making major design decisions without having accurate information on the eventual system characteristics. This paper presents a novel constraint-driven methodology to support system-level design. The software assists a designer or a tool in partitioning behavioral specifications onto multiple VLSI chips and in system design while satisfying hard constraints such as individual chip areas, chip pin counts, system throughput (inverse of system initiation interval) and system latency (delay). The software uses search and estimation techniques to perform comprehensive design-space exploration and evaluates partitions supplied by the user or by other synthesis software. The technique determines what design characteristics each partition must possess in order to satisfy area, pin, throughput and latency constraints. The paper also includes a brief description of an estimation tool used in this methodology and the results from experiments performed for validation.

Keywords— High-level Synthesis, System-level Design, System-level Synthesis, Partitioning, Estimation, Prediction, Design-space Exploration.

1 Introduction

In the search for a good system design, there are many tradeoffs which have to be examined, and the mutual dependence between the tradeoff criteria and eventual physical design characteristics make the many system-level design decisions guesswork. A major difficulty at the system level is that many decisions made at this level are binding throughout the subsequent stages of the design. Many architectural assumptions which have been used in the past to make system-level design decisions do not hold for current technologies due to decreases in feature sizes and technology impacts. This forces consideration of physical design impacts early in the design process and results in an increased need to search a larger portion of the design space to find the best design. Even though automated synthesis can speed up the design process by orders of magnitude, the growing complexity of designs together with the multitude of tradeoffs possible during synthesis make it impossible to completely search the design space using fast synthesis tools. The designer has to iterate several times by modifying the functional specification (high-level transformations), changing design constraints and making new design decisions. Furthermore, most synthesis tools solve complex subproblems and do not take a global view of the problem being solved.

To further complicate the design process, many digital designs are too large to fit on a single chip. At present, much chip partitioning is performed manually by designers. With the increasing complexity of designs and quick turn-around time requirement, fast partitioning mechanisms must be used.

The time and the way partitioning is performed are quite important. The quality of synthesis results is highly sensitive to many interacting tradeoffs and design decisions. Therefore, if a design were synthesized as if it were a single chip design, but were later partitioned onto multiple chips, the chances of synthesizing a globally inferior design could be high due to the fact that some early design decisions were made without information about partition boundaries. Scheduling operations into time steps imposes both a minimum and maximum time constraint on the clock cycle. The minimum clock-cycle time exists due to the longest execution delay assigned to any time slot. The maximum clock-cycle time exists due to global throughput and latency constraints, and the schedule length in terms of time slots. If the partitioning were performed after synthesis, it might not be possible to find any feasible[1] partitioning due to the introduction of off-chip communication delays which may violate maximum clock-cycle constraints. The introduction of off-chip communication delays after synthesis might only be resolved by stretching the clock-cycle time which could violate the global throughput or latency constraints. Such delays might arise due to pin limitations which force multiplexing, pad, and off-chip wire delays. In such a case, the synthesis process would somehow have to be repeated a number of times on a trial basis to find a feasible partitioning.

We describe here a novel constraint-driven system design tool CHOP whose methodology is able to determine the feasibility of tentative partitions at the behavioral level by using accurate estimation techniques as it makes system-level design decisions. This partitioning evaluation methodology enables a designer to search a much larger portion of the design space in a limited time compared to conventional methods.

In short, the problem we are solving is the following: Our interest is in the system synthesis of data-path dominated designs which focuses on module selection, scheduling, resource allocation and sharing, and interconnect and controller synthesis. A behavioral specification

[1]Feasible: Satisfying all area, pin, throughput, and latency constraints.

for an arithmetic computation in the form of a Data-Flow Graph (DFG)[2] or a program from which a DFG could be extracted is used an input. Examples (with no conditional constructs) of each representation are shown in Figures 11 and 13, respectively. Section 3.2 further describes the types of descriptions currently handled by the software. In the rest of the paper, *behavioral specification* and *DFG* will be used interchangeably. The designer wishes to explore the design space including single and multiple die implementations. The designer has also determined that dies will communicate synchronously with identical clock rates. Finally, the designer (or software) has tentatively divided the behavioral specification into initial partitions (which most likely reflect functional boundaries) and assigned these partitions to dies with selected package types. Our methodology determines whether there is adequate space on each die to contain all partitions assigned to the die and additional logic for the off-chip data transfers, and also whether there are adequate pins available for data transfers, both of which must be determined in the face of system-wide timing constraints on throughput and latency. The timing of each partition must be compatible with other partitions so that the flow of data between partitions is feasible. If there is more than one partition on a single chip, the combined area characteristics of these partitions have to satisfy the area constraint of the chip. In a globally-feasible design, faster non-pipelined implementations of partitions can be used along with slower pipelined implementations of others. Pipelining between non-pipelined partitions is considered as well. The solution process determines a design style (pipelined or non-pipelined) and a module set for each partition as well as predicted area and timing behavior of each partition.

The organization of the paper is as follows. Section 2 discusses related research in behavioral partitioning. Section 3 discusses various aspects of CHOP's system design approach. Experimental results are given in Section 4. Summaries of the novel single-chip behavioral area and delay estimation techniques and the prototype tool BEST, which was primarily developed for use in CHOP are given in the Appendix. More detailed information on the system design methodology can be found in [10]. Early work on the methodology was reported in [9].

2 Related Research

Most related research involves automatic partitioning. McFarland partitions behavioral specifications with a clustering algorithm based on a *similarity* measure [15]. These clustering algorithms are employed in allocation and module binding phases of the data-path synthesis process. A behavioral partitioning technique developed by Lagnese and Thomas based on McFarland's work uses a multi-level clustering approach to generate the best intuitive partitioning which improves the quality of single-chip designs [13]. Their results show significant area reductions, but their approach does not consider design constraints.

A general and widely used heuristic for partitioning graphs with costs on the edges into subgraphs of specified sizes while trying to minimize the total cost of edges cut was invented by Kernighan and Lin [8]. This heuristic has been applied successfully to partitioning of logic circuits and Register-Transfer Level (RTL) designs.[3] At the RT or logic level it is rel-

[2]DFG: A directed graph whose nodes and edges represent operations and values, respectively. The graph used here is typical of those used in high-level synthesis. A tutorial which provides the necessary background to the reader on high-level synthesis concepts, terminology, specification methods and existing systems can be found in [17].

[3]RTL Design: Structural design at the architectural component level, e.g., word-level adders, incre-

atively easy to use accurate measures to evaluate the *quality* of the partitioning because the subsequent changes to the structure of the implementation after partitioning are minimal. On the other hand, at the behavioral level, global information about the final design characteristics useful to partitioning is non-existent since no structure exists. High-level synthesis also introduces significant and generally irregular sequential behavior into the design. This causes most final design characteristics to be a function of the sequential behavior introduced and hence a function of the structure produced as well as of the original behavior. Without having the results of high-level synthesis or accurate estimations of these results, it has not been shown if one can directly correlate "sum of costs of values cut" to the pin count requirement or "sum of sizes of operations in a partition" to the area of chips.

A graph-theoretical behavioral partitioning technique has been reported by Gupta and De Micheli [4]. The designer manually finds a starting partitioning which satisfies the timing constraints, and a preliminary schedule for the design is produced, then the Kernighan-Lin algorithm or simulated annealing is used to finalize the partitioning. Each operation has an abstract area. The authors do not consider pin-sharing or area/delay characteristics of registers, multiplexers, controllers, or wiring.

The CAMAD [22] system takes a different approach. CAMAD uses an extended timed Petri-net model as the behavioral level specification. Costs derived from the Petri net represent the importance of the data/control connectivity and these costs are assigned to places, transitions and edges connecting them to merge the partitioning of data-path and control constructs. Then, Kernighan-Lin type of algorithms are used to perform the partitioning.

Vahid and Gajski reported a specification partitioning tool at the process/procedural level [26] which tries to find feasible partitions with respect to chip-pin count, chip area, and performance using well-known partitioning techniques (e.g., [8, 16]). Estimations for the area and the performance, and chip-pin constraints are used to drive the partitioning. Area and delay estimations of super-nodes (e.g., a process) are obtained by a combination of synthesis and estimation techniques (for area) and by simulation (for delay). Resource sharing between the processes, serial/parallel tradeoffs, the effects of multiplexing, control and routing on the performance, and migration of operations between the processes are not considered.

Gebotys and Elmasry incorporated a partitioning model into an integer programming formulation for high-level synthesis [2]. This formulation simultaneously performs partitioning, scheduling, and allocation, and considers off-chip delays and off-chip buses, but does not model the effects of multiplexing, control or routing on area or performance.

3 The System Design Approach

CHOP is a system design tool which provides an evaluation mechanism to determine the quality and feasibility of a partitioning quickly. CHOP helps a designer or another tool partition behavioral specifications onto multiple chips while satisfying hard physical constraints. CHOP also determines design styles, module styles, and desired cost and performance for each partition.

CHOP introduces a new methodology (set of methods) for multi-chip system design. In this methodology the designer (or another tool) maps behavioral specifications in the form of a graph to multiple chips, subject to area, throughput and latency constraints and decides

menters, ALUs, registers, and register files.

on chip packaging options before any implementation exists. A tool like CHOP assists the user by providing extensive estimations and fast design-space exploration capabilities so that the designer can quickly determine the best achievable global characteristics (e.g., throughput and latency). Then, the user can drive the data-path synthesis tools according to CHOP's directions to obtain the predicted implementation with minimal design iteration. This methodology promotes the use of accurate cost functions throughout the design, unlike most partitioning efforts described in Section 2. First, it provides the user with accurate bottom-up feedback so that system-level design tasks can be performed with accurate cost functions. Secondly, it allows the user to define important system characteristics of the design (e.g., partitioning and target chip set) early in the design cycle so that subsequent design tools can optimize the design with more accurate cost functions.

The partitioning model supports single/multi-chip implementations with multiple partitions per chip. The reason we support multiple partitions per chip is that it has been shown by Lagnese and Thomas that dividing a single-chip design into multiple partitions without sharing any hardware between the partitions is likely to reduce the routing area and can produce better designs [13]. The partitioning model also allows non-uniform partitioning of the behavior onto chips. By allowing for variations in target chip sizes and packaging, CHOP increases the chances to reach a low-cost multi-chip implementation. This cost is not a direct function of the number of chips or the total area, but rather is a function of the costs of actual dies used as well as packaging.

The system design approach is comprehensive. Detailed estimation techniques for area and timing addressing most digital design aspects are included. The design and estimation techniques include consideration of design style selection (pipelined/non-pipelined), module style selection, operator, register and multiplexer prediction, wiring area/delay prediction, controller area/delay prediction, determination of memory bandwidths, determination of off-chip communication bandwidths, performance matching between independent data paths, determination of data buffering, and architecture of a distributed control mechanism. CHOP uses probabilistic methods to determine the feasibility of tentative partitions, based on these estimations and design decisions.

3.1 Inputs to CHOP

The input data required for CHOP can be summarized as follows: the behavioral specification in the form of a data-flow graph, a library of components, the chip set onto which the design is to be partitioned, on- and off-chip memory modules to be used, assignments of memory modules to chips, partitions, assignments of partitions to chips, tentative data-path and data-transfer clock-cycle times, the architecture style, and the feasibility criteria.

The library generally consists of more than one component which can implement each operation type. It is assumed that the memory hierarchy is designed prior to partitioning although, in practice, designers interleave iterations of memory architecture design and behavior partitioning, a step now automated in ADAM [21]. The chip-set information is in the form of actual chip packages to be used. The information about each chip includes the dimensions of the project area, the pin count of the chip, pad delays, and I/O pad area. Off-chip wire delays are currently assumed to be lumped into pad delays. The architecture style, selected by the user, can have single-cycle or multi-cycle operations, and should be compatible with the architecture style of the synthesis tools which will later be used to implement the hardware. Approximate data-path and data-transfer clock-cycle times which

consider only functional delays are inputs to the system. CHOP later estimates more accurate clock-cycle times taking into account estimated register, multiplexer, PLA, wiring and chip-to-chip delays. The feasibility criteria consist of constraints for chip area, throughput and latency, and specification of how strictly each constraint will be enforced (explained in Section 3.3.5).

3.2 Assumptions and Limitations

CHOP is intended for data-path intensive applications rather than control-dominated systems. A typical data-path intensive application area is Digital Signal Processing. Pipeline flushing is not considered in our throughput measures. A distributed control is assumed for the partitioned design in which finite-state machines communicate with each other synchronously. Due to the fact that the data-processing rate can be different from the chip-to-chip data-transfer rate, we assume two separate clocks for data paths and data transfer, both of which are derived from the same master clock. In order to keep design complexity at a moderate level, both clocks in our model are to be synchronized with periods which are integer multiples of the master clock period. In an actual implementation, this model may require 1-3 clocks. CHOP's design methodology supports and encourages hierarchical designs, but the prototype tool itself does not support hierarchy, to simplify the implementation. The partitioning modeling discussed in the paper does not yet take into account any potential off-chip switching elements which may be needed as a result of the partitioning. These switching elements may be required to couple and de-couple off-chip buses.

The are two sets of estimation work reported in this paper; one set of estimations related to multi-chip design (CHOP), another set related to single partition data-path synthesis (BEST). BEST (Behavioral Area-Delay ESTimator) is the first proof-of-concept prototype which performs estimations from behavior to layout. It was primarily developed to supply potential implementations for each partition to CHOP. Since the partitioning prototype currently uses the BEST tool to estimate the characteristics of single partition designs, it currently inherits assumptions and limitations of BEST.[4] For example, BEST assumes that all operations have deterministic delays. The effects of hardware chaining on the clock-cycle time are not considered. The effects of conditionals on the allocation estimations are not yet as accurate as we wish. In addition to inherited limitations from BEST, CHOP imposes a topological restriction on the partitioning to assure the accuracy of the estimations: Any two partitions should not have *mutual* dependencies. Details on this can be found in [10].

CHOP's methodology can handle loops although the prototype tool itself does not. This is due to the fact that BEST cannot handle inner loops at this time. Most designs can be classified into two groups; Real-time systems and non-critical systems. Real-time systems cannot work with unbounded delays (including *while* loops). Non-critical systems with unbounded delays are generally defined with expected throughput and latency. Since CHOP's methodology works with numbers for throughput and latency, independent of the fact that these numbers are absolute or expected, it is possible to extend BEST, hence CHOP to handle unbounded delays.

[4]Details on BEST could be found in the Appendix or in [10]. The use of BEST in CHOP is explained in detail in Section 3.3.

3.3 CHOP Operation

The overall operation of CHOP is shown in Figure 1. CHOP first obtains a number of possible implementations for each partition by calling BEST. CHOP then uses a combination of search, synthesis and estimation techniques to construct the best global design from these partition implementations. During this process, system-level issues such as feasibility checks and the overhead to integrate partitions into a system are taken into account. As a final step, the designer (or a tool) modifies the system-level design (DFG, partitions, memory architecture, target chip set, and timing constraints) based on the feed-back from CHOP and iterates until a satisfactory design is achieved.

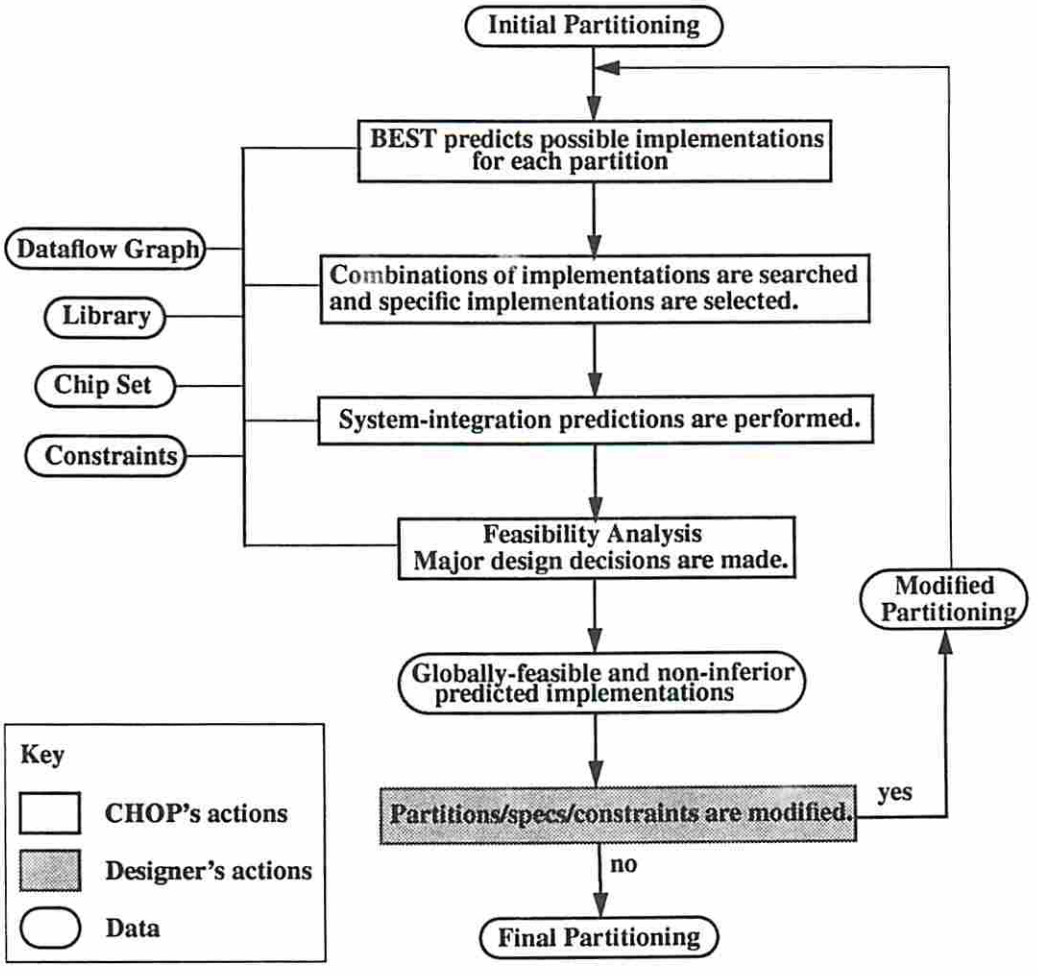

Figure 1: The overall operation of CHOP.

3.3.1 CHOP Overview

An overview of CHOP's operation will be given this section. This overview will be followed by more detailed discussions in the subsequent sections. The partitioning methodology in

CHOP can be best explained by an example. An example tentative partitioning consisting of 5 partitions ($P1 - P5$), and 2 memory units (M_A and M_B) as a four-chip design is shown in Figure 2. It is important to note the following: there can be multiple partitions which may or may not have dependencies on others assigned to a single chip, but no two partitions should have mutual dependency on each other (e.g., no loops). Memory blocks can be assigned to the same chips as partitions, but the use of off-the-shelf memory chips is also allowed. In the same fashion, any pre-designed implementation of partitions (off-the-shelf data-path parts) can also be used.

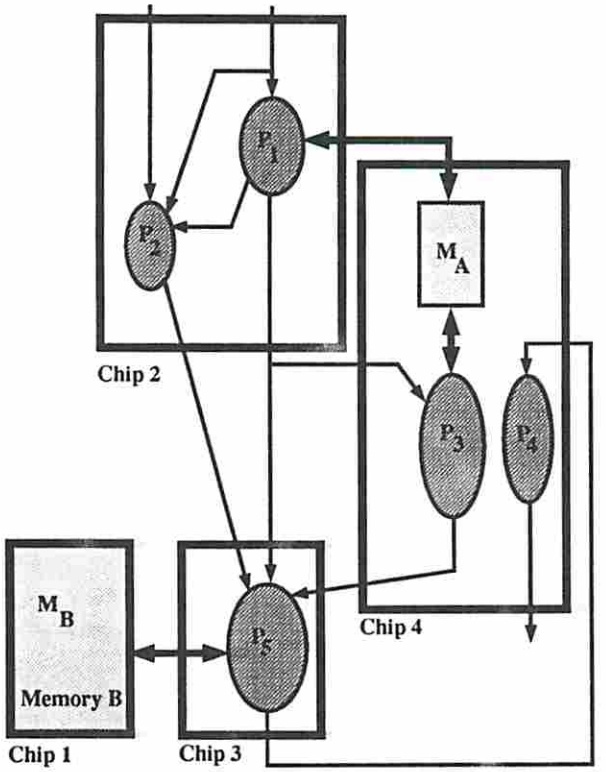

Figure 2: An example partitioning.

The first step in the partitioning process is the creation of a task graph as shown in Figure 3. The task graph is a directed acyclic graph with nodes representing partitions and off-chip data transfers and edges representing the flow of data. Creation of the task graph involves determining the off-chip data broadcast scheme (single or multiple destination broadcast) and the amount of data to be transferred, and reserving enough pins for signals which cannot share pins (e.g., control signals to assure proper communication between distributed controllers and select signals for memory blocks). CHOP assumes that a separate hardware unit will be used to implement each task (i.e. processing units for partitions and data-transfer modules for data-transfer tasks). The architectural building blocks shown in Figure 4 are to be used to implement the partitioning shown in Figure 2, resulting in the partitioned implementation shown in Figure 5.

After the creation of a task graph, CHOP determines possible individual implementations for each partition by calling BEST. The results from BEST include global characteristics for area, throughput, latency and memory bandwidth requirements for each memory block due to memory accesses from inside the partition.

CHOP next explores how to construct multi-chip system implementations from estimated

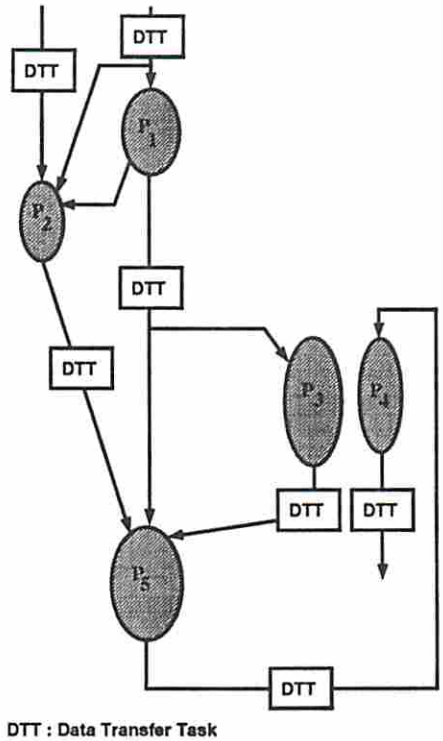

Figure 3: The task graph for the example partitioning.

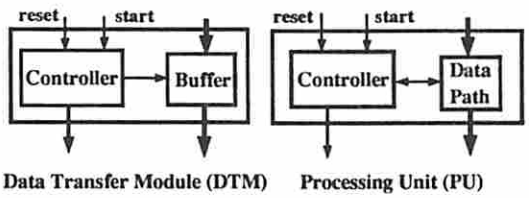

Figure 4: The architectural building blocks.

individual partition implementations. When multiple implementations with different area-delay characteristics can be produced for each partition (as shown for a 3-partition design in Figure 6), one implementation must be selected for each partition to construct the best global implementation while satisfying global design constraints.

Global implementations with very different design characteristics can be obtained for each combination of individual partition implementations. Since the system-integration overhead[5] is highly dependent on the way the global implementation is constructed, CHOP attempts to accurately estimate the global design characteristics by exploring possible combinations of partition implementations followed by the system-integration estimation for each combination explored. Each estimated global implementation is then put through the feasibility analysis to find out if it satisfies the design constraints.

[5]System-integration overhead includes area and delays associated with buffers, data-transfer controllers and chip pin multiplexing; and off-chip delays.

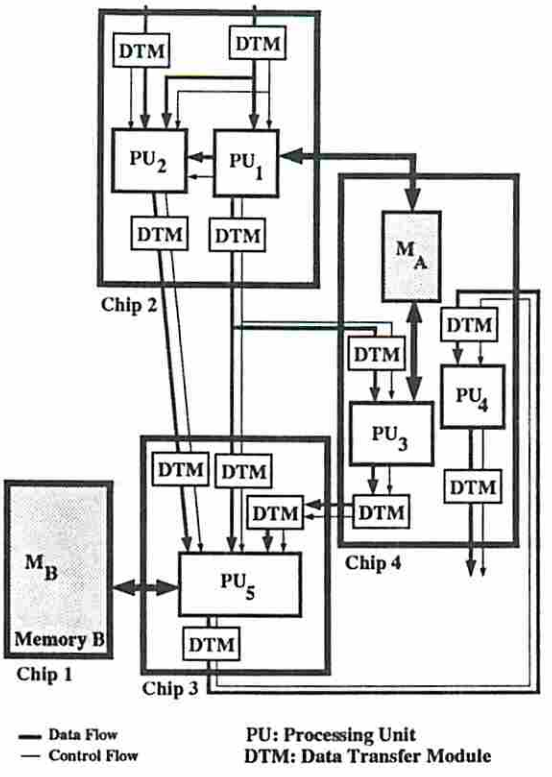

Figure 5: The final implementation of the example partitioning.

3.3.2 Generation of Partition Implementations

CHOP currently uses the BEST predictor to populate the implementation space for each individual partition, characteristics of pre-existing implementations (design reuse support) or results from actual synthesis can be used instead of performing estimations. Using synthesis to populate the implementation space for individual partitions might be very slow, but it may be tolerable for partitions which are not modified frequently.

BEST estimates the area-delay trade-off curves of potential designs which can be generated by a typical data-path synthesis system. It explores different design styles, library configurations (each library configuration contains one operator per operation type), and serial/parallel trade-offs. The results from BEST include global characteristics such as area, throughput, latency, and memory bandwidth requirements, and more detailed estimates such as predicted operator, register and multiplexer allocation, predicted PLA-based controller area (including the area of registers storing the state bits), and predicted standard-cell routing area, as well as the predicted delays introduced into the clock cycle (register, multiplexer, PLA and wiring delays). Techniques used in BEST are summarized in the Appendix.

3.3.3 Search Mechanism for Partition Implementation Selection

Before invocation of any search, partition-implementation choices can be examined to eliminate the ones which cannot lead to any feasible global implementation. The early elimination of such partition implementations contributes to the reduction of run time and has no effect on the partitioning results. In our methodology, there are two types of elimination techniques to reduce the size of the search space. The first type of elimination is independently performed on each partition implementation by subjecting it to throughput, latency, and

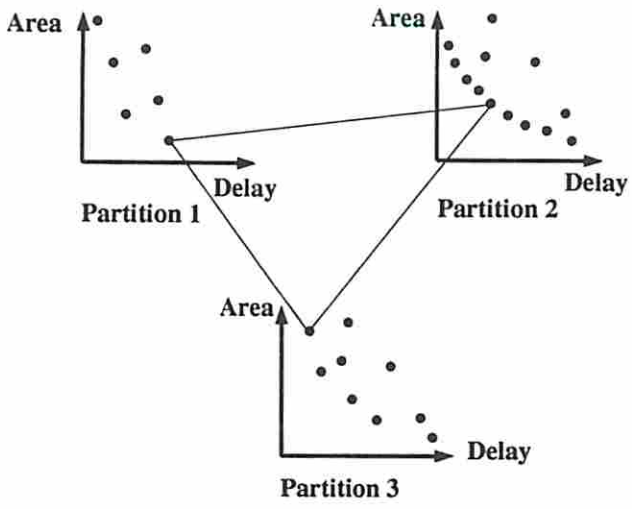

Figure 6: Selection of partition implementations.

chip area constraints. Inferior[6] partition implementations are also eliminated at user's option, but this may affect the partitioning results. The second type of elimination takes into account the combined characteristics of partitions and it is described by the following simple theorems:

Theorem 1 *Let i vary over the partitions on a chip. The implementation area A_k of partition k to be used in a feasible global design is upper bounded as follows:*

$$A_k \leq \text{Chip Area Constraint} - \sum_{i \neq k} A_{i,min}$$

where $A_{i,min}$ is the minimum implementation area of partition i.

Theorem 2 *Let $D_{i,k}$ be the delay of the k^{th} implementation of partition i. Let $D_{i,min}$ be the minimum implementation delay of partition i, which is given as*

$$D_{i,min} = \min_{k} (D_{i,k})$$

Let $CP_{i,k}$ be the critical-path delay for the global design using $D_{i,min}$ for all partitions except partition i, and $D_{i,k}$ for partition i. If

$$CP_{i,k} \geq \text{Global Delay Constraint}$$

then, the k^{th} implementation of partition i cannot be used in any feasible global design.

The proofs for the above theorems are omitted since they are trivial and are given in [10]. Theorem 1 states that the area of any partition implementation is upper bounded by the maximum area that partition can occupy on its assigned chip. This maximum area is achievable when other partitions on the same chip are implemented with the smallest possible area. Given a selected implementation of a partition, Theorem 2 defines the minimum system (global) latency achievable. If this achievable system latency is not feasible then the selected implementation can be eliminated without affecting the partitioning results.

[6]Designs which have no better but at least one worse characteristic for the criteria considered than at least one other design are considered inferior.

The user has the option of using either of two exhaustive search techniques or a heuristic search technique built into CHOP to find the best global implementation which can be constructed from several possible implementations for each partition. One of the exhaustive search techniques explores multiple ways to pipeline each combination of partition implementations. Both exhaustive search techniques are computationally intensive with long run times for realistic problems and have been implemented to check the quality of the heuristic explained below.

The heuristic search technique in CHOP tries to find a global implementation with the highest system throughput (first priority) and the shortest system latency (second priority) while satisfying chip area and pin constraints. The heuristic starts with the fastest-throughput implementation for each partition and iteratively considers more serial/slower implementations of partitions residing on chips whose area constraints are violated. Selection of more serial implementations is done in such a way that the increase in system latency caused by serialization is minimized. This selection generally favors the serialization of off-critical-path partitions (off-critical with respect to the global implementation considered at the time). The outline of the algorithm is given in Figure 7.

3.3.4 System-Integration Overhead Estimates

System-integration estimates include data-transfer-module characteristics and the throughput and latency characteristics of the overall system. Since the number of pins of each chip is pre-determined, data-transfer times are functions of the amount of data to be transferred and the availability of pins. It is assumed that the maximum possible bandwidth is used for each data transfer, and each data-transfer task and partition should wait until its entire required bandwidth (for pins and memory) is available.[7] During bandwidth calculations, the effects of simultaneous off-chip memory access on pin usage are also taken into account. This bandwidth is then used to calculate the duration (delay) of each data-transfer task.

By fixing the resource requirements (bandwidth) and delays of data-transfer tasks and partitions in this way, resource-constrained scheduling techniques can be used to check the feasibility. CHOP performs an urgency scheduling of all tasks to check the feasibility of sharing the data pins of chips and memory blocks while reaching the minimum overall system latency. The urgency measure is based on the critical-path delays of tasks and is similar to the urgency measure used in [20]. The delay of the resulting task schedule is used as the estimated system latency.

The resulting task schedule is also used in estimating the characteristics of data-transfer modules. For each data-transfer task, one data-transfer module has to be placed on each of the chips involved in the data transfer (one in the output mode, the rest in the input mode as shown in Figure 5). Each output data-transfer module may contain a *wait* until enough pins are available for the data transfer, followed by the data transfer and each input data-transfer module performs a data transfer which may be followed by a wait time to hold the data until the destination partition(s) can accept the data. The modes of operation for the data-transfer tasks are shown in Figure 8.

Although individual pipelined and non-pipelined partition implementations may co-exist in the final design, the overall process (including the data-transfer tasks) is always assumed to be pipelined. For example, consider the partitions on Chip 1 in Figure 2. For a system

[7] Of course, the final implementation might produce more complicated pin allocation and hence better performance than CHOP estimates.

initiation interval of 3 clock cycles, partition $P1$ can be a pipelined implementation with the initiation interval of 3 clock cycles and latency of 4 clock cycles while partition $P2$ can be a non-pipelined implementation with latency of 2 clock cycles. In another words, partition $P1$ finishes its computation quicker than partition $P2$ can accept data from partition $P1$. CHOP considers how to design a system using these partitions and their implementations. This includes generating the task schedule ($P2$ executing after the completion of $P1$) and estimating buffers to hold primary (external) inputs of partition $P2$ until the time needed. The wait time for a data-transfer task could be longer than the initiation interval of the system, resulting in additional data buffering. On the other hand, the data-transfer duration for each data-transfer task cannot be longer than the initiation interval of the system in order not to cause data clashes on the pins since pin counts are hard user constraints which are not altered by CHOP. Each data-transfer module is active from the time there is a start signal for new data transfer until the time the data-transfer module relinquishes control of the data to the destination partition(s). The sufficient buffer size, B needed for each data-transfer module is estimated as

$$B = \text{Data Size} \times \lceil \frac{\text{Wait} + \text{Transfer Delay}}{\text{initiation interval}} \rceil$$

where "Wait" is obtained from the task schedule produced by CHOP.

Buffers between partitions which are on the same chip are considered similarly, except that there is no off-chip data transfer involved.

The number of inputs, outputs and product terms of PLAs used to control the data transfers are estimated from the tasks' wait and execution times. Then, PLA sizes and delays are estimated using methods similar to those in BEST. The area and delays of registers storing the state bits of these controllers are also estimated.

Estimation techniques assume that all data pins are shared I/O pins. Off-chip accesses are required due to off-chip memory accesses and off-chip data transfers. Given a specific implementation for each partition, the number of off-chip memory ports (and therefore address and data lines) used by each partition implementation is known. The total bitwidth of values being transferred off-chip is also available from the partitioning information.

Let O_i be the total number of bits to be transferred off chip i. Then, the number of 1-bit 2-to-1 multiplexers required to share γ_i data pins of chip i, $muxcnt_{pin}$ is estimated as

$$muxcnt_{pin} = \max(O_i - \gamma_i, 0)$$

The delays due to pin multiplexing, $muxdelay_{pin}$ are estimated as

$$muxdelay_{pin} = \lceil log_2 \max(\frac{O_i}{\gamma_i}, 1) \rceil \times \text{2-to-1 mux delay}$$

If there are off-chip memory accesses from partitions, then pin multiplexing introduces delays into both data-path and data-transfer clocks. If pins are used only for data transfers between partitions, then the estimated delay is added only to the data-transfer clock.

Procedure find_feasible_partitioning_implementations
Let W_i be an estimated implementation of partition i.
Let L_i be the initiation interval of W_i.
Let S be the list of candidate partitions for serialization.
Sort all W_i for each partition i in increasing order, first for the initiation interval,
then for the system latency, and finally for the area.
for each feasible initiation interval l
  for each partition i
    Initialize W_i to the first (fastest) estimated implementation
    in the sorted estimation list of partition i.
  Advance each W_i until $L_i \geq l$ or W_i is a non-pipelined implementation with $L_i \leq l$.
  while TRUE
    If $\exists$ any W_i not implementable with initiation interval l
      break
    Estimate system integrations using l and W_is.
    If the estimation result is feasible
      Record the overall estimation results.
      Set S to the list of all partitions.
    else
      Set S to partitions on chips whose area constraint
      is violated by the last system integration estimation.
    Move W_i of each partition on S forward to skip the estimation points
    which have inferior characteristics or are not implementable with l.
    For each partition on S
      Move W_i forward (serialize) on its list.
      Find the expected system latency using the urgency
      scheduling using chip pins and memory blocks as resources.
      Restore the old value of W_i.
      Record the partition for which the serialization resulted in the minimum system latency.
    If $\exists$ a partition recorded above
      Serialize the partition. (Move W_i one element forward in its list)
    else
      Serialize a partition arbitrarily.
End.

Figure 7: The Outline of the Iterative Heuristic for Partition Evaluation.

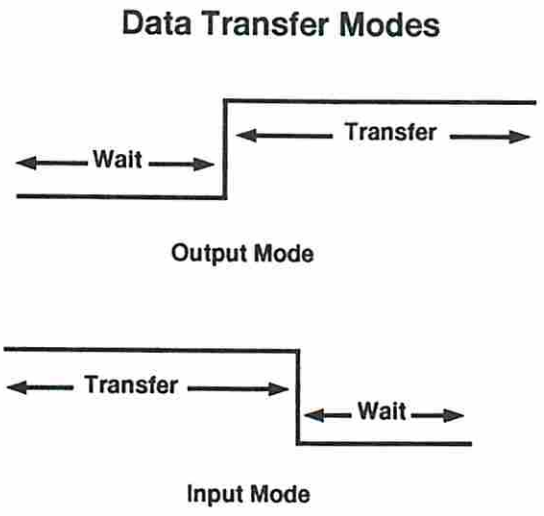

Figure 8: The Modes of Operation for Data-Transfer Tasks.

3.3.5 Feasibility Analysis

The feasibility of the partitioning is checked when the individual characteristics of processing units and data-transfer modules have been determined. The feasibility analysis is performed for each chip area constraint by considering the area taken by processing units, data-transfer modules residing on each chip, and multiplexing to share the data pins of each chip. The throughput and latency characteristics of PUs, chip-packaging delays, chip-to-chip delays, controller delays and pin-multiplexing logic delays are used to calculate the delay introduced into the tentative master clock-cycle time. First , the master clock-cycle time is stretched to accommodate these additional delays, then the feasibility of the throughput and the system latency are checked.

The master clock cycle is then recalculated as follows: let k_{dp} (k_{xfer}) be the ratio of data-path (data-transfer) clock-cycle times to the master clock-cycle time. Also let d_{dp} (d_{xfer}) be the sum of delays introduced into the data-path (data-transfer) clock-cycle time. Then, the adjusted master clock-cycle time, c' is given as

$$c' = \max\left(c + \frac{d_{dp}}{k_{dp}}, \frac{d_{xfer}}{k_{xfer}}\right)$$

where c is the initial clock-cycle time estimate given to CHOP. This clocking model allows minimal interference between the data-path and data-transfer clock rates without increasing the complexity of the clocking.

Performing feasibility checks during automatic design-space exploration using estimations require special care. Since each estimation result have a different degree of uncertainty associated with it, performing feasibility checks strictly based on a simple arithmetic comparison (e.g., $1000 > 999.99$) could lead to unjustified elimination of potential designs from consideration. Therefore, it is important to model the uncertainty factors in estimation results so that statistical techniques can be used to make more accurate decisions.

Both CHOP and BEST prototypes use the PERT modeling technique [14, 23] to model the uncertainties in the estimation results. Each estimation result is generated with three

values: a lower-bound, a most likely and an upper-bound value.[8] These values represent a statistical distribution and are used to compute an average and a variance for each estimation result. Then, the average and the variance are used with standard statistical methods [24] to perform the feasibility analysis.

For example, the PERT modeling technique is used for the area characteristic. *area.lb*, *area.ml*, *area.ub* are the estimated lower-bound, most-likely and upper-bound values of an area component (e.g., operator area) respectively. Based on these three values, the PERT model approximates the average value for area and variance of area (*area.avg* and *area.var*) as follows (see Figure 9):

$$area.avg = \frac{area.lb + 4 \times area.ml + area.ub}{6}$$

$$area.var = \frac{(area.ub - area.lb)^2}{36}$$

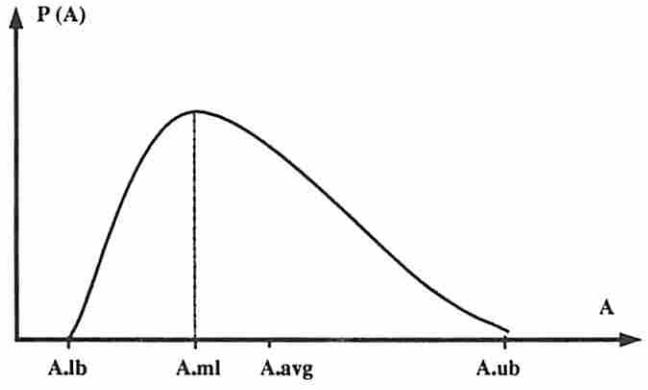

Figure 9: The PERT model used in estimations.

Average total area is computed by summing individual area averages. Total area variance is computed by summing individual area variances.

After aggregate estimation results for a particular design characteristic have been computed, infeasible estimated designs are eliminated as follows. The PERT model states that aggregate characteristics obtained from PERT-generated distributions can be approximated with normal distributions. The accuracy of these approximations depends on the data-pool size and how well the PERT numbers fits the application. Therefore, the accuracy of estimations are still important. The probability of satisfying the hard area constraint A_{max} for the estimated aggregate area A is computed as the area under the probability density curve for A to the left of A_{max}, as shown in Figure 10. The shape of the probability density curve for A is dependent on the aggregate average and variance. Any estimated design whose probability of satisfying a constraint is lower than the user-specified minimum probability of feasibility for the same constraint is eliminated. This computation is readily available in a tabular form which makes it suitable for fast execution [24].

Using the PERT model, both CHOP and BEST can model different uncertainty factors for individual estimation techniques (low, high, and most-likely values) as well as how strictly

[8]These bounds are statistically significant bounds and not necessarily absolute theoretical bounds.

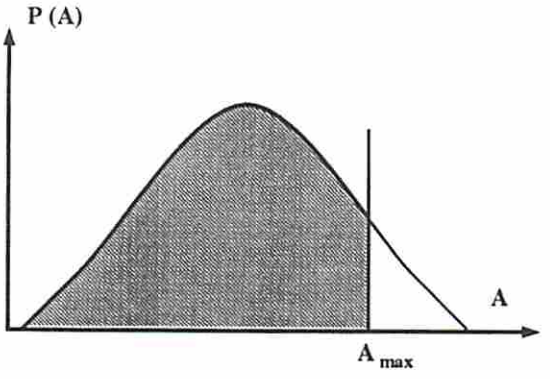

Figure 10: The probability of feasibility as the area under a probability density curve.

each global constraint should be enforced (user-specified probability of acceptance). Using a single value for the low, high, and most-likely values for a particular characteristic completely removes the uncertainty factor from the characteristic since the variance is calculated as zero. If all estimation results were in single-value form then the aggregate results in BEST and CHOP would not have any variance and constraints would be applied without consideration to any uncertainty.

3.3.6 Partitioning Modification

After the feasibility of a partitioning is checked, the partitioning process can be continued as the designer/tool modifies the specifications and constraints, based on the feedback from the partitioner. Modifications can be grouped into 4 major groups:

Behavioral Partitions: The changes can be as simple as operation migrations from partition to partition, or migration of partitions from chip to chip. It is also possible to decrease or increase the number and the size of partitions.

Memory blocks: The assignments of memory blocks can be changed to modify the number of off-chip memory accesses.

Target chip set: The feasibility of behavioral partitioning is highly dependent on the target chip set. Less area available for the design generally translates into slower, more serial designs. By the same token, the fewer the number of pins available for data transfer, the longer the data transfer takes. Modifications in target chip-set characteristics affect the feasibility of the behavioral partitioning. Target chip characteristics generally dictate the overall manufacturing cost of the design.

Timing constraints: High system throughput and short system latency constraints translate into more parallelism in the feasible designs, which are larger in size. High throughput constraints also cause the I/O pin usage to increase. Relaxing unnecessarily tight constraints is also a good way to increase the number of feasible designs.

During the modification steps above, the goal of the designer or the system-level design tool should be to achieve

- the maximum parallelism within the partitions and among the partitions,
- the best partition granularity for the design (if partition granularity is too coarse or too fine the chance of synthesizing a good quality design decreases),
- the maximum chip-area utilization and the maximum chip-pin utilization,

- the maximum operator and register utilizations within partitions (e.g., implementing 3 additions on 2 adders in 2 clock cycles does not utilize one of the adders for one of the clock cycles), and
- the best balanced data-path and off-chip data-transfer clock-cycle times.

3.4 Outputs from CHOP

During its search CHOP constructs many tentative designs, but only outputs feasible non-inferior designs. Virtually all design decisions and individual estimation results for these reported designs such as the partitioning, system pipelining frequency, style of transferring data between the chips, chip-pin multiplexing, coarse scheduling of tasks, the decided design style, the selected module styles, the decided parallelism for scheduling (e.g., 2-step non-pipelined schedule), and operator, register, interconnect, and routing estimations for each partition are available from CHOP.

Most data-path synthesis tools utilize external constraints to guide their optimizations. The detail of these constraints vary greatly from global constraints such as total area, throughput, or latency to more specific constraints such as the clock cycle time or the total operator area, to even more specific constraints such as complete operator and register allocation. The design decisions and estimations results supplied by CHOP can be used to drive the synthesis tools. A typical use of synthesis directions from CHOP by a designer is as follows:

- Take a partition of the design,
- use the subset of the library as suggested,
- use pipelined or non-pipelined scheduler as suggested,
- give the needed constraints to the scheduler (e.g., 4 time-step schedule, clock cycle time c, n multipliers, m adders.),
- monitor deviations in synthesis results from the estimated characteristics, and if any synthesized characteristic seems inferior explore the possibility of further improvement.

The use of CHOP's synthesis directions is a fully manual process at this time.

3.5 Complexity

The run-time complexity of CHOP's evaluation of a multi-chip system design depends on

- the complexity of pre-processing,
- the complexity of populating individual partition-implementation spaces (currently the complexity of BEST),
- the total number of individual partitioning implementations,
- the search method used, and
- the complexity of system integration estimations.

The run-time complexity of CHOP's evaluation using BEST and the heuristic search technique is upper bounded by

$$\mathcal{O}(\rho mn^2 + mn \log m + \rho mn(t^2zt + z^3))$$

where ρ is the number of partitions, m is the number of library configurations, n is the number of operations, t is the number of tasks (dependent on partitioning topology), and z is the number of possible initiation intervals (dependent on timing constraints and the number of operations in each partition). Data on actual run times is given in Section 4.3.

4 Experimental Results

In this section, a summary of the experiments which were performed to validate the CHOP tool and partition evaluation methodology will be described. Since the accuracy of CHOP's results highly depend on the accuracy of estimations given to CHOP by BEST, a short summary of validation experiments for the BEST tool is also given at the end of the Appendix. More experimental data both on CHOP and BEST is available in [10].

4.1 Partitioning Experiments

Several experiments have been performed with the interactive partitioner. The first set of experiments was performed to validate the estimation and design techniques used in CHOP, and the results are given in Section 4.1.1. The next set of experiments includes several partitioning evaluations and explorations using the interactive partitioner, and some of the results are presented in Section 4.1.2. Based on the results of these experiments, the actual run times of CHOP and the variation of results depending on the search method used in CHOP are discussed in Section 4.3. A simple automatic 2-way partitioning technique is discussed in Section 4.4 to show how the partitioning evaluation by CHOP can be used within a higher-level technique. I/O-Bound designs are discussed in Section 4.5 using a Discrete Cosine Transform (DCT) example.

4.1.1 Validation of System Design

The validation of the partitioning estimations was performed by implementing the partitions and by comparing the estimation results to actual synthesis results. One partitioned specification for each of the AR and FIR Filters was synthesized by the ADAM synthesis tools to validate the estimations used by CHOP. During the synthesis of partitions CHOP's synthesis directions (see Section 3.4) were used directly. The interface hardware (buffers and multiplexing logic) was manually designed. Manually-written finite-state-machine specifications for the interface hardware were used generate the data-transfer PLAs using the UC Berkeley PLA synthesis package (PEG, EQNTOTT, ESPRESSO, and MKPLA) [5].

Partitioning estimations and synthesis were performed subject to constraints on throughput, latency, chip areas, and chip pin counts. CHOP's area-constraint enforcement was set to be tight. This was done so that the synthesis for the estimated design would have a greater chance of satisfying the chip area constraints. System-design experiments were performed using the 3-micron library in Table 1 which allowed 9 library configurations.

The partitioning shown in Figure 11 and the partitioning shown in Figure 12 were used to validate the accuracy of the partitioning estimations. Each partition shown in Figure 11

			3 micron		1.2 micron	
Module Name	Type	Bit Width	Area (mil^2)	Delay (ns)	Area (mil^2)	Delay (ns)
add1	Adder	16	4200	34	126.0	35.0
add2		16	2880	53		
add3		16	1200	151		
mul1	Multiplier	16	49000	375	3118.0	54.5
mul2		16	9800	2950		
mul3		16	7100	7370		
register	Register	1	31	5	9.3	3.0
mux	2:1 Mux	1	18	4	3.0	3.6

Table 1: The library used in the experiments.

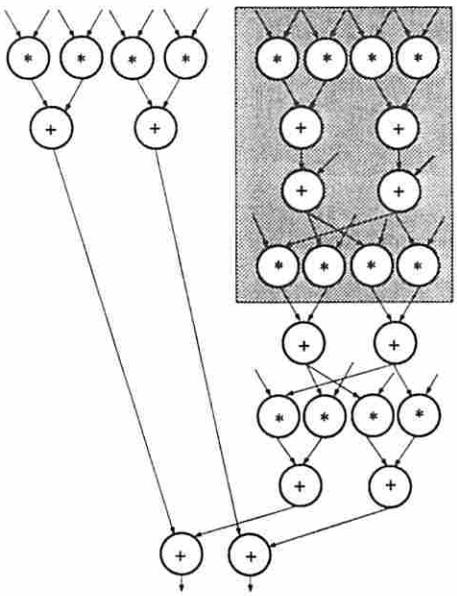

Figure 11: A 2-way partitioning of the AR Filter Element.

and Figure 12 was assigned to a separate chip. Type-C packages from Table 4 were used in these experiments. Minor changes were made to synthesis results to ensure highly optimized results when our older ADAM Synthesis tools missed obvious choices. The final design characteristics followed the estimations closely as shown in Tables 2 and 3. There are no actual layout areas reported in Tables 2 and 3 since it was not possible for us to generate layouts for the 3-micron technology library used at the time of the experiments. However, the layout-related estimation techniques in BEST and CHOP were directly imported from Kurdahi's work and were previously shown to have an accuracy of 10 percent using actual layouts [12]. Therefore, it is sufficient to compare the estimated and the actual active area for validation purposes.

4.1.2 Partitioning Evaluations

A set of experiments has been performed using CHOP interactively to investigate how much throughput increase can be obtained by effective partitioning of behavioral specifications. The results for the AR and FIR Filters are given in Tables 5 and 6. In the experiments,

	Estimated	Synthesized
Initiation Interval (clock cycles)	20	20
System Latency (clock cycles)	45	45

Area in mil^2	Chip No	Estimated	Synthesized
Processing Unit Active Area	1	58940	53770
Interface Active Area	1	14936	12344
Total Active Area	1	73876	66114
Total Area	1	130279	N/A
Processing Unit Active Area	2	55811	56314
Interface Active Area	2	23030	22897
Total Active Area	2	78841	79211
Total Area	2	137031	N/A

Active area includes RTL and PLA area.
The total area includes standard-cell routing area for processing units.

Table 2: Estimated and Synthesized Design Characteristics of the 2-way partitioning of the AR Filter shown in Figure 11.

both single-partition and multi-partition implementations were considered. CHOP's area constraint enforcement was set to be tight. The throughput and latency constraints were both set loosely so that CHOP would report the best implementation which satisfies the area and pin count constraints. The tentative data-path and data-transfer clock-cycle times (to be later adjusted by CHOP) were set to 3,000 and 300 ns. The full 3-micron library which allowed 9 library configurations was used. Unless otherwise noted, in all 1, 2 or 3 chip implementations, Type-B chip packages were used from Table 4. Except for experiment 4 in Table 5 and experiment 5 in Table 6, every partition was assigned to a separate chip.

As it can be seen from the data reported in Table 5, the single partition implementation of the AR Filter had to be highly serial due to chip area constraints. By introducing another chip, the throughput can be increased over 4 times with marked improvement in the overall system latency. Such a big difference in the throughput between the 1-chip and the 2-chip AR

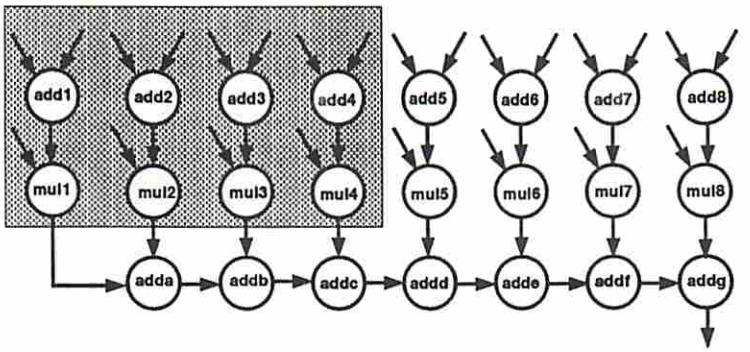

Figure 12: A 2-way partitioning of the FIR Filter.

	Estimated	Synthesized
Initiation Interval (clock cycles)	20	20
System Latency (clock cycles)	36	35

Area in mil^2	Chip No	Estimated	Synthesized
Processing Unit Active Area	1	60604	52839
Interface Active Area	1	7000	6484
Total Active Area	1	67604	59323
Total Area	1	114202	N/A
Processing Unit Active Area	2	30279	33178
Interface Active Area	2	17079	14518
Total Active Area	2	47358	47696
Total Area	2	96964	N/A

Table 3: Estimated and Synthesized Design Characteristics of the 2-way partitioning of the FIR Filter shown in Figure 12.

	Project Area (mil)				Single Pad Area
Package	Width	Length	Pad Delay	Pin Count	(mil^2)
A	271.65	267.72	25.0	40	297.60
B	311.02	362.20	25.0	64	297.60
C	311.02	362.20	25.0	84	297.60
D	311.02	362.20	25.0	120	297.60

Table 4: The partial list of MOSIS standard chip packages [19]. Package type D, a hypothetical package which does not exist in the MOSIS library, was introduced for the experiments reported in Section 4.5.

implementations is mainly due to 2 factors; the 2-chip implementation simply allowed more fine-grain parallelism within the partitions, but also allowed coarse-grain pipelining between the partitions. Adding the third chip still improves the throughput with some small penalty on the system latency over the 2-chip implementation. In almost all cases, the estimated chip area usage was above 85%.

As it can be seen from the data reported in Table 6, the throughput of the FIR Filter can be increased more than twice by partitioning the design onto two chips without any penalty on the system latency. Adding the third chip still doubles the throughput and improves the system latency with respect to the 2-chip implementation. The 4-partition, 3-chip implementation results in maximum throughput and minimum latency since it allows maximum parallelism between the partitions. In almost all cases (with the exception of experiment 3), the estimated chip area usage was above 80%.

Multiplexing, controller, wiring and pad delays ranging from 40 to more than 120 ns were introduced into the data-path clock cycle for the experiments reported in Tables 5 and 6. However, the impact of delays on the master clock cycle was minimal due to the fact that the delays introduced into one data-path clock cycle were actually distributed over 10 master clock periods. In designs with fast data-path clock rates, impacts of such delays would be

Experiment	P	C	Initiation Interval (clock cycles)	System Delay (clock cycles)	Master Clock-Cycle Time (ns)
1	1	1	168	168	309
2	2	2	40	67	309
3	3	3	20	67	308
4	3	2	30	58	309

P : Number of partitions
C : Number of chips

Table 5: AR Filter Tentative Designs.

Experiment	P	C	Initiation Interval (clock cycles)	Delay (clock cycles)	Master Clock-Cycle Time (ns)
1	1	1	48	48	313
2	2	2	20	46	309
3	3	3	20	39	311
4	3	3	10	46	305
5	4	3	10	27	305

Table 6: FIR Filter Tentative Designs.

considerably higher.

4.2 Non-uniform Partitioning in CHOP

Since the target chip characteristics are related to the manufacturing cost of the design, if the throughput and latency constraints allow, it is preferable to search for lower-cost implementations which may be achievable only by a non-uniform partitioning. An example is given in Table 7. The prices of chips for packages A and B are \$305 and \$467, respectively. An extensive search was performed for 2-partition 2-chip implementations of the AR Filter onto different target configurations of the chip sets as shown in Table 7. As it can been seen from the table, a range of multi-chip designs differing in cost and performance can be produced with non-uniform partitioning.

Design	Package Type		Price	Initiation Interval (clock cycle)	Delay (clock cycle)
	Chip 1	Chip 2			
1	B	B	\$934	40	67
2	B	A	\$774	50	99
3	A	A	\$610	90	172

Table 7: Different cost 2-chip implementations of the AR Filter using MOSIS chip packages.

4.3 Comparison of Search Methods

The three techniques currently implemented in CHOP for the global implementation search perform the search differently, so, the results as well as the run-times can be quite different. The statistical information and run times on the experimental results presented in Tables 5 and 6 are given in Tables 8 and 9, respectively. All CPU times reported are on a Sun Sparc 4/460 in seconds. CPU times quote the computation time and do not include I/O and pre-processing times.

To date, the iterative heuristic found the same results as the better exhaustive-search technique with orders of magnitude speed advantage. The reduced exhaustive search technique failed to find as good estimates as the other techniques for some of the evaluations reported.

		# of global implementation trials		
Experiment	N_p	ES 1	ES 2	Heuristic
1	207	39	378	83
2	207	1681	22965	38
3	228	42315	622153	19
4	206	18522	273321	18

N_p : Total number of estimations produced by BEST
ES 1 : Exhaustive Search 1
ES 2 : Exhaustive Search 2 (with better results)

	CPU Time (*sec.*)		
Experiment	ES 1	ES 2	Heuristic
1	0.32	1.06	0.44
2	4.13	34.02	0.41
3	68.19	789.85	0.36
4	25.84	236.44	0.34

Table 8: Statistical data on AR filter evaluation results in Table 5.

4.4 A Simple Automatic 2-way Partitioning

CHOP provides a good infrastructure and excellent means for accurate evaluation of a partitioning. In order to demonstrate how easily and effectively higher-level automatic partitioning techniques can be implemented using CHOP, its framework and its accurate cost functions (estimations), a simple automatic 2-way partitioning technique for horizontal cuts have been implemented on top of CHOP. The additional coding to implement this automatic partitioning technique on top of CHOP took only one day. The technique starts the search with all operations assigned to one of the chips. Then, it moves the operations one by one to the other chip and evaluates each partitioning using CHOP. The technique tries two different orders (ASAP and ALAP) for moving operations and also tries different orderings of the chip assignments, if the chip packages are different. The technique finally reports the best partitioning encountered. Although the automatic 2-way partitioning technique is very

		# of global implementation trials		
Experiment	N_p	ES 1	ES 2	Heuristic
1	201	75	820	35
2	198	2016	28980	19
3	203	49348	700408	20
4	156	454272	7239576	20

	CPU Time (*sec.*)		
Experiment	ES 1	ES 2	Heuristic
1	0.37	1.58	0.35
2	3.96	35.64	0.27
3	71.77	722.68	0.36
4	361.99	3644.47	0.26

Table 9: Statistical data on FIR filter evaluation results in Table 6.

simple, it is quite successful in finding a good partitioning since it uses CHOP's accurate cost functions.

This simple automatic 2-way partitioning technique was used on the AR Filter and the FIR Filter. The experiment set up was the same as the one assumed in Section 4.1.2. For each design, the resulting partitioning was the same as the best partitioning previously found by extensive interactive partitioning. These were reported as experiment 2 in Table 5 and experiment 2 in Table 6.

The automatic 2-way partitioning of the AR Filter and FIR Filter took 54 and 44 partition evaluations, respectively. The total run times of each case (using the heuristic) were 31.43 and 16.33 seconds, respectively.

CHOP performs a brute-force search in the design space. The amount of search performed by CHOP can be reduced drastically by using tighter constraints. Elimination of some operators from the library which are too large or too slow for the design goals and constraints also proves to be very useful. Our experience shows that more than an order of magnitude speed-up can be achieved by carefully constraining the search. The automatic 2-way partitioning experiment uses the partitioning evaluation several times. When the throughput constraint was tightened and BEST was asked not to generate any estimations for pipelined data paths (the library was kept unchanged), the total run time for each case dropped to 3.93 and 3.5 seconds, respectively, without any change in the results. This shows the importance of tightly constraining the search performed by CHOP.

4.5 I/O Bound Designs - A DCT Example

Arbitrary partitioning of some behaviors onto multiple chips is sometimes severely constrained by the I/O bandwidth of multi-chip partitioning rather than the chip areas. The I/O boundedness of a multi-chip design is affected by the amount of off-chip accesses including ones which are added due to the partitioning, the amount of pins available for these off-chip accesses, the ratio of data-path and data-transfer clock-cycle times, off-chip data-transfer delays (e.g., pad delays), the library used, the design methodology and style.

The DCT description shown in Figure 13, which was originally taken from [1], will be used to demonstrate some I/O-bound partitioning cases. We explored 1 and 2-chip implementations of the DCT for a variety of chip packages from Table 4 using CHOP. This experiment used the 1.2 micron library from Table 1, 55 *ns* seed clock-cycle time for data-path and data-transfer clocks, and CHOP's automatic 2-way partitioning technique for 2-chip implementations. As it can be seen from the results of the experiment shown in Table 10, adding more real estate (chip area in the form of an additional chip) of type B or C did not help to improve the throughput of the design. In these cases, a significant portion of the chip areas were not utilized due to I/O the boundedness of the implementations. Only a 2-chip implementation with hypothetical D-type packages which have 120 pins increased the throughput. The examination of the DCT data-flow graph and additional manual partitioning experiments using CHOP strengthened our belief that DCT was not easily partitionable without having substantial I/O bandwidth available.

There are generally two solutions to improve the throughput of I/O bound designs: either make more pins available, and/or allow the creation of more efficient and balanced I/O patterns, which can be achieved by including the I/O behavior in the original behavioral description, and let the synthesis tools schedule the I/O operations as well. Revising the algorithm in this case is desirable, since many iterations are possible using the fast prediction/decision tools implemented in CHOP and BEST.

5 Conclusion and Future Research

We have demonstrated a behavioral system-level design methodology and an interactive prototype tool, CHOP. CHOP is the first tool, to our knowledge, which facilitates the partitioning of behavioral specifications onto multiple chips while trying to satisfy physical design constraints such as chip areas, pin counts, throughput, and latency. CHOP (including BEST) is coded in C and is approximately 12,000 lines of code. CHOP explores the design-space for single or multi-chip design styles and provides quick feedback to the designer on what characteristics can be expected from the implementation of the design.

CHOP gives a previously unavailable design-space exploration capability to the designer. CHOP explores $\sim$ 100 designs per CPU second. Since its search is done intelligently, the effective design space covered in such a search is equivalent to $\sim$ 1000 to 10000 designs. To synthesize a layout from an algorithmic description of similar sizes to those presented in the paper would take at least a day. The speed-up CHOP brings into design-space exploration over generating trial designs is six to seven orders of magnitude. This previously non-existing design-space exploration capability allows a designer to search for several alternative implementation schemes prior to proceeding with the detailed implementation. The experiments show that significant area/cost/performance/design-time benefits can be obtained using the proposed approach without paying a comparable price.

CHOP can be used easily by a system designer to check the effects of system-level decisions in real time. We feel that arbitrary automatic system-level behavioral partitioning is not appropriate in many multi-chip VLSI designs since system-level partitioning is also affected by factors not discussed in this paper such as design reuse (non-partitionable macros), designer preference to keep partitions meaningful w.r.t. some other criteria, and modular testability/simulation concerns.

```
...
 ...
entity dct is
  port(in0,in1,in2,in3,in4,in5,in6,in7, a,b,c,d,e,f,g : in SixteenBitVector;
       out0,out1,out2,out3,out4,out5,out6,out7 : out SixteenBitVector);
end dct;

architecture behavior of dct is
begin process
  variable t1,t2,t3,t4, m1,m2,m3,m4,m5,m6,m7,m8 : SixteenBitVector;
  begin
        t1 := in0+in7;          t2 := in3+in4;
        t3 := in1+in6;          t4 := in2+in5;

        m1 := t1+t2;            m2 := t3+t4;
        m3 := t1-t2;            m4 := t3-t4;
        m5 := in0-in7;          m6 := in1-in6;
        m7 := in2-in5;          m8 := in3-in4;

        out0 <= d*(m1+m2);      out4 <= d*(m1-m2);
        out2 <= b*m3 + f*m4;    out6 <= f*m3 - b*m4;

        out1 <= a*m5 + c*m6 + e*m7 + g*m8;
        out3 <= c*m5 - g*m6 - a*m7 - e*m8;
        out5 <= e*m5 - a*m6 + g*m7 + c*m8;
        out7 <= g*m5 - e*m6 + c*m7 - a*m8;
  end process;
end behavior;
```

Figure 13: VHDL [27] description for the 8-point 1D-DCT.

					Area Usage (%)	
Package Type	Number of Partitions and Chips	II	SD	MC	Chip 1	Chip 2
B	1	11	11	99	85	-
B	1	7	14	99	87	-
B	2	11	12	99	78	25
B	2	7	14	99	80	25
C	1	10	10	99	90	-
C	1	6	12	99	92	-
C	2	10	11	99	90	22
C	2	6	12	99	92	22
D	1	5	10	97	96	-
D	2	4	10	99	94	40

II : Initiation Interval (clock cycles)
SD : System Delay (clock cycles)
MC : Master Clock-Cycle Time (ns)

Table 10: Results for the DCT example.

There are a number of open problems in system-level design automation. These include estimation techniques with better accuracy, estimation techniques covering more design methodologies, more detailed and explicit consideration of actual design costs, hardware-software co-design in both estimations and synthesis, and automatic partitioning. The ultimate goal of this work is to be able to perform system-level tasks for hardware and software simultaneously, to trade off between hardware and software, to trade off between different hardware implementation methodologies, and to produce the least "cost" board-level design satisfying constraints including power consumption and reliability.

6 Acknowledgements

The authors would like to thank Pravil Gupta and Atul Ahuja for producing the layouts and for their help in synthesizing some of the designs. Atul Ahuja also converted some of BEST technology parameters from 3-micron to 1.2-micron technology. Pravil Gupta provided the DCT VHDL description. This work was supported in part by the Defense Advanced Research Projects Agency and monitored by the Federal Bureau of Investigation under contract No. JFBI90092.

References

[1] H. Fujiwara, M. L. Liou, M. T. Sun, K. M. Yang, M. M. Maruyama, K. Shomura, and K. Ohyama. An All-ASIC Implementation of a Low Bit-Rate Video Codec. *IEEE Trans. on Circuits and Systems for Video Technology*, 2(2):123–134, June 1992.

[2] C. H. Gebotys and M. I. Elmasry. Optimal Synthesis of Multichip Architectures. In *Proc. Int'l Conf. on Computer-Aided Design*, pages 238–241. IEEE, November 1992.

[3] P. Gupta. PLA Delay Analysis. Wire Delay Analysis. Department of Electrical Engineering, University of Southern California, 1990. Internal Reports.

[4] R. Gupta and G. De Micheli. Partitioning of Functional Models of Synchronous Digital Systems. In *Proc. Int'l Conf. on Computer-Aided Design*, pages 216–219. IEEE, November 1990.

[5] G. Hamachi. Designing Finite State Machines with PEG. Technical report, University of California, Berkeley, 1983.

[6] R. Jain, K. Küçükçakar, M. J. Mlinar, and A. C. Parker. Experience with the ADAM Synthesis System. In *Proc. 26th Design Automation Conf.*, pages 55–61. ACM/IEEE, June 1989.

[7] R. Jain, A. C. Parker, and N. Park. Predicting System-level Area and Delay for Pipelined and Non-pipelined Designs. *IEEE Trans. on Computer-Aided Design*, 11(8):955–965, August 1992.

[8] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 49(1):291–307, January 1970.

[9] K. Küçükçakar and A. C. Parker. CHOP: A Constraint-Driven System-Level Partitioner. In *Proc. 28th Design Automation Conf.*, pages 514–519. IEEE/ACM, June 1991.

[10] K. Küçükçakar. *System-Level Synthesis Techniques With Emphasis On Partitioning And Design Planning.* PhD thesis, Department of Electrical Engineering, University of Southern California, October 1991.

[11] S. Y. Kung, H. J. Whitehouse, and T. Kaliath. *VLSI and Modern Signal Processing.* Prentice-Hall, 1985.

[12] F. J. Kurdahi and A. C. Parker. Techniques for Area Estimation of VLSI Layouts. *IEEE Trans. on Computer-Aided Design*, 8(1):81–92, January 1989.

[13] E. D. Lagnese and D. E. Thomas. Architectural Partitioning for System Level Synthesis of Integrated Circuits. *IEEE Trans. on Computer-Aided Design*, CAD-10(7):847–859, July 1991.

[14] R. I. Levin and C. A. Kirkpatrick. *Planning and Control with PERT/CPM.* McGraw Hill, 1966.

[15] M. C. McFarland. Computer-Aided Partitioning of Behavioral Hardware Descriptions. In *Proc. 20th Design Automation Conf.*, pages 472–478. ACM/IEEE, June 1983.

[16] M. C. McFarland. Using Bottom-Up Design Techniques in the Synthesis of Digital Hardware from Abstract Behavioral Descriptions. In *Proc. 23rd Design Automation Conf.*, pages 474–480. ACM/IEEE, June 1986.

[17] M. C. McFarland, A. C. Parker, and R. Camposano. The High-Level Synthesis of Digital Systems. *Proc. IEEE*, 78(2):301–318, February 1990.

[18] M. J. Mlinar. *System Level Tradeoffs in VLSI Design.* PhD thesis, Department of Electrical Engineering, University of Southern California, May 1991.

[19] MOSIS User Manual. USC-Information Sciences Institute, 1990.

[20] N. Park and A. C. Parker. Sehwa: A Software Package for Synthesis of Pipelines from Behavioral Specifications. *IEEE Trans. on Computer-Aided Design*, 7(3):356–370, March 1988.

[21] A. C. Parker, Chih-Tung Chen, and Pravil Gupta. Unified System Construction. In *Proc. Fourth SASIMI Workshop*. Nara, Japan, October 1993.

[22] Z. Peng. Synthesis of VLSI Systems with the CAMAD Design Aid. In *Proc. 23rd Design Automation Conf.*, pages 278–284. ACM/IEEE, June 1986.

[23] A. Ravindran, D. T. Phillips, and J. J. Solberg. *Operations Research: Principles and Practice. Second Edition.* John Wiley and Sons, Inc., 1987.

[24] E. A. Robinson. *Probability Theory and Applications.* Int'l Human Resources Development Corporation, 1985.

[25] Seattle Silicon Corporation. *ChipCrafter Designer's Handbook*, March 1990.

[26] F. Vahid and D. D. Gajski. Specification Partitioning for System Design. In *Proc. 29th Design Automation Conf.*, pages 219–224. ACM/IEEE, June 1992.

[27] IEEE Standard VHDL Language Reference Manual. The Institute of Electrical and Electronics Engineers Inc., March 1988.

Appendix

A BEST (Behavioral Area-Delay ESTimator)

BEST is a comprehensive and integrated area-delay estimation tool to support system-level design. The overall data-path synthesis process (down to the layout) is modelled as a design decision tree as shown in Figure 14, the leaves of the decision tree being the potential designs which can be generated. BEST generates estimations for all possible meaningful implementations of the design in its framework which is taken from the ADAM System [6]. But, the estimations are more general and not designed for use exclusively by ADAM. All possible library configurations (each library configuration contains one operator per operation type) are enumerated. For each library configuration, pipelined and non-pipelined estimations are generated for estimated ranges of the initiation interval (in terms of clock cycles) and the number of stages. Each estimation task contributes some estimated characteristics of each potential design to the data pool and uses the original input data as well as estimation results obtained prior to its execution. The allocation, control generation and layout tasks of design are currently modelled with a single style each. If there were multiple styles considered for those tasks, there would be corresponding branches in Figure 14.

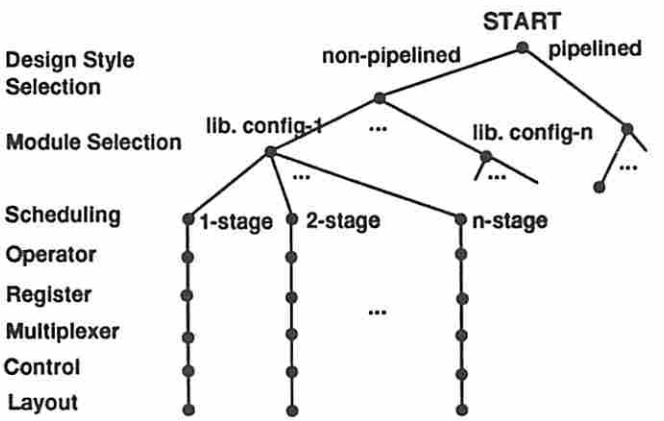

Figure 14: The current estimation generation in BEST.

BEST has a total of 22 estimation techniques built-in, 9 of which are inherited either as is, with modification, improvement, or simplification from previous USC research, and 13 are newly developed. These estimation techniques include the following: maximum fine grain parallelism across conditional operations [20] (e.g., maximum 8 out of 12 conditional operations can execute in parallel), critical-path delay (calculation with an approximate model – no bit-accurate hardware chaining), serial-parallel tradeoff range (min and max) consisting of the range for the number of time steps (non-pipelined design style) and the range for the throughput (pipelined design style), number of time steps as a function of the throughput (pipelined design style), number of operators of each type [7], dataflow graph width, number of registers as a function of parallelism [18], register delays, average operation chaining as a function of parallelism, complexity of operator multiplexing as a function of parallelism, complexity of register multiplexing as a function of parallelism, total multiplexer area, multiplexer delays introduced into the clock cycle, abstract controller parameters (product terms,

inputs, and outputs) [18], PLA area for the controller [18] (imported as is), PLA delay for the controller [3] (imported as is), wiring complexity (number of nets and fanout), average wire length [12] (imported as is), individual wire delays [3] (imported as is), number of wires in sequence within time step boundaries, effect of wire delays into the clock cycle, and finally routing area [12].

A summary of techniques used in BEST and some experimental results will be given in the following sections. The notation used in the Appendix is defined in Table 11.

N	The number of stages in the schedule.
C	The critical-path delay with a given library.
l	The initiation interval in terms of clock cycles.
n_i, o_i	The number of operations (operators) of type i.
a_i, d_i	The area (delay) of the operator of type i.
a_r, a_m	The area of a 1-bit register (2-to-1 multiplexer).
n_i^e	The maximum number of operations of type i which can be executed in parallel. n_i^e is equal to n_i for designs with no conditionals and it is less than or equal to n_i for design with conditionals [20].
ibw_i, obw_i	The total input (output) bitwidth for operation/operator type i.
I, O	The total number of external input (output) bits.
I_c	The total number of constant input bits.
Y	The total output bits of all operations.
A	The average bitwidth of operations in the dataflow graph.
R_{lb}, R_{ml}, R_{ub}	The estimated number of bits of registers (lower-bound, most-likely, upper-bound).
M_{lb}, M_{ml}, M_{ub}	The estimated number of bits of 2-to-1 multiplexers.
T	The maximum number of stages considered for the design style.
N_c	The number of operations on the critical path.
$\lceil \frac{N}{l} \rceil$	The number of data sets simultaneously executed in the pipeline.
$\frac{N_c}{N}$	The average operation chaining.
R_{max}	The estimated graph width (in terms of bits) which is the estimated maximum number of single-bit registers in any non-pipelined implementation.
$d(wl, f)$	The RC delay through a wire of length wl and fanout f. Refer to [10] for details of the connection model.
op_mux, reg_mux	The largest operator (register) multiplexer tree size.
$\alpha, \beta, \gamma, \kappa, \tau$	Experimental constants for the ADAM System.

Table 11: Notation used in the Appendix.

A.1 Range Estimation for Serial/Parallel Tradeoffs

Theorem 3 *Assuming that there are no externally imposed internal timing constraints, the serial/parallel tradeoff range for non-inferior non-pipelined designs, N is bounded by*

$$\lceil \frac{C}{c} \rceil \leq N \geq \sum_i n_i \times \lceil \frac{d_i}{c} \rceil$$

where i varies over operation types.

Since this upper bound is a loose bound, the following inequality is used in BEST to determine the useful serial/parallel tradeoff range.

$$\lceil \frac{C}{c} \rceil \leq N \leq \max\left((\sum_i n_i \times \frac{d_i}{c}), \max_i (n_i \times \lceil \frac{d_i}{c} \rceil) \times \kappa \right)$$

Theorem 4 *Assuming that there are no externally imposed internal timing constraints, serial/parallel tradeoff range for non-inferior pipelined designs, l is bounded by*

$$1 \leq l \leq \max_i \left(n_i \times \lceil \frac{d_i}{c} \rceil \right)$$

The proofs of Theorems 3 and 4 are given in [10].

A.2 Operator Allocation Estimation

Operator allocation estimations are derived from Jain's original work [7]. Jain's original work has been improved to handle multi-cycle operations. Given the design style, module library, and the parallelism (from the scheduling phase) lower-bound operator allocation is calculated by the following theorems:

Theorem 5 *Given n_i^e, d_i, c, l, and the pipelined design style, the sufficient number of operators of type i in the design is given as*

$$o_i \geq \lceil \frac{n_i^e \times \lceil \frac{d_i}{c} \rceil}{l} \rceil$$

Theorem 6 *Given n_i^e, d_i, c, N, and the non-pipelined design style, the sufficient number of operators of type i in the design is given as*

$$o_i \geq \lceil \frac{n_i^e \times \lceil \frac{d_i}{c} \rceil}{N} \rceil$$

The proofs for Theorems 5 and 6 are given in [10].

In the pipelining model currently used, it is always possible to achieve the lower-bound operator allocation at the expense of longer pipelines; therefore, the lower-bound, the most-likely and the upper-bound values for o_i are all the same. The most-likely and the upper-bound operator allocations in non-pipelined designs are currently estimated in a heuristic manner.

A.3 Register Estimation

The estimation of register area is derived from Mlinar's work [18]. Mlinar first finds the exact width of the data-flow graph but we currently use a simple heuristic. This is followed by the estimation of the number of registers as a function of the parallelism (including pipelining), the graph width and other graph characteristics. The outline of the technique is

Procedure Register Estimation
if $N > l$ **then** {Pipelined}
$R.lb = \min(I - I_c, O) + \lceil \frac{N}{l} - 1 \rceil \times R_{max}$
$R.ub = \max(I - I_c, R_{max}) \times \lceil \frac{N}{l} \rceil$
$R.ml = \frac{R.lb + R.ub}{2}$
else {Non-pipelined}
if $N = 1$ **then**
$R.ml = O$ {Only outputs are stored}
else
if $N > T$ **then** $N = T$ {Saturate N}
$R.ml = \lceil \frac{N-1}{T-1} \rceil \times \max(R_{max} - O, 0) + \alpha \times O$ {Linear interpolation}

$R.ub = \max(O, R_{max})$
$R.lb = O$ {At least outputs are stored.}
If inputs need to be stored
$R.ml = \max(R.ml, I - I_c)$
$R.ub = \max(R.ub, I - I_c)$
$R.lb = \max(R.lb, I - I_c)$
End.

Figure 15: Register Area Estimation Technique.

shown in Figure 15. $R.ml$ for the non-pipelined designs can be anywhere between $R.lb$ and $R.ub$. We have not seen any characterizable pattern for the number of registers within these bounds from the synthesis results. Currently, the linear interpolation intentionally favors more parallel implementations. This turns out to be very useful in automatic selection of the best design since serial designs are penalized unless there is a significant area gain from the serialization. Register delays are two times the register propagation delay due to non-overlapped execution of data and control paths in our model.

A.4 Interconnect Logic Estimation

The effects of multiplexing is considered by first estimating the complexity of operator and register sharing as a function of allocation. This is followed by the estimation of multiplexer area and delays. The total multiplexer area and delay estimation techniques have empirical constants derived from experiments. These constants may need to be derived again if there is a drastic change in the synthesis methodology or if estimations are to be ported onto another synthesis system. The outline of the technique is shown in Figure 16.

The calculation of $muxcnt1$ assumes that inputs of all operations are unique, which gives a loose theoretical upper bound on the number of operator multiplexers. The calculation of $muxcnt2$ assumes that there is a single multiplexing network for all operator multiplexing and estimates the number of multiplexers in the network from the number of inputs and outputs of the network.

The aggregate multiplexer delays introduced into the clock cycle are calculated with the assumption that signals will travel through one set of multiplexers for registers and $\lceil \frac{N_c}{N} \rceil$ sets

Procedure Multiplexer Estimation

$muxcnt1 = \sum_i ibw_i \times (n_i - o_i)$ {Operator multiplexing}

$muxcnt2 = I + R.ml + \sum_i o_i \times (obw_i - ibw_i)$ {Operator multiplexing}

$muxcnt = (muxcnt1 + muxcnt2)/2$ {Average of two methods}

$muxcnt = muxcnt + \max(Y + I - R, 0)$ {Register multiplexing}

$M.lb = \beta \times muxcnt \times$ 2-to-1 mux area {Empirical Observation}

$M.ml = \gamma \times muxcnt \times$ 2-to-1 mux area {Empirical Observation}

$M.ub = muxcnt \times$ 2-to-1 mux area {Empirical Observation}

$op\_mux = \max_i(n_i - o_i)$ {Largest operator mux tree}

$reg\_mux = \max(Y + I - R, 0)/A$ {Largest register mux tree}

$avg\_delay = (\lceil \log_2(\frac{reg\_mux}{\tau}) \rceil + \lceil \frac{N_c}{N} \rceil \times \lceil \log_2(\frac{op\_mux}{\tau}) \rceil) \times$ 2-to-1 mux delay

$max\_delay = (\lceil \log_2 reg\_mux \rceil + \lceil \frac{N_c}{N} \rceil \times \lceil \log_2 op\_mux \rceil) \times$ 2-to-1 mux delay

if $l = 1$ **then**

  $M, op\_mux, reg\_mux = 0$ {No multiplexing}

  $avg\_delay, max\_delay = 0$ {No multiplexing}

$md.lb = 0$ {Multiplexing delay}

$md.ml = avg\_delay$ {Multiplexing delay}

$md.ub = max\_delay$ {Multiplexing delay}

End.

Figure 16: Interconnect Area and Delay Estimation Techniques.

of multiplexers for operators (due to operation chaining). For average delay estimations it is assumed that the multiplexer use in an actual design will be a fraction of the maximum values calculated.

A.5 Control Estimation

The control estimator assumes a PLA implementation. First abstract PLA characteristics (the number of PLA inputs and outputs, and the number of product terms) are estimated in a similar way to that proposed by Mlinar [18]. Then, Mlinar's [18] and Gupta's [3] techniques are used to estimate the area and the delay of the PLA implementation, respectively. Since PLA delays are transition dependent, they are characterized by three charging-discharging behaviors of PLA planes; best-best, average-average, worst-worst.

A.6 Wiring Estimation

Wiring estimations are currently based on the standard-cell layout methodology. The wire area estimation is taken from Kurdahi [12]. Kurdahi's area estimation technique requires the RTL area and the equivalent number of 2-point nets as inputs. These are estimated as follows:

$$A_{RTL} = R \times a_r + M \times a_m + \sum_i o_i \times a_i$$

$$nets = \left(\sum_i o_i \times ibw_i\right) + R + 2 \times M + O$$

where i varies over operator types.

We performed a series of experiments and showed that it was possible to interpolate the wiring area using previous results generated by Kurdahi's technique. Since run-time was critical for us, we adopted a table look-up interpolation technique based on previous results. This table look-up interpolation technique gives results within 1% of Kurdahi's original results.

Wire delays are estimated as a function of the maximum and the average multiplexer complexities for operators and registers, the average fanout, the average operation chaining, and the average wire length. The average fanout is estimated as the ratio of total number of module inputs over the total number of module outputs (including primary inputs and outputs).

$$avgf = \frac{\left(\sum_i o_i \times ibw_i\right) + R + 2 \times M + O}{\left(\sum_i o_i \times obw_i\right) + R + M + I}$$

The upper bound on the aggregate wire delay introduced into the clock cycle is estimated as the delay through a average-length wire with average fanout multiplied by the maximum number of wires on a clock-to-clock critical path.

$$wd.ub = d(avg\_wl, avgf) \times \left(1 + \lceil \log_2 reg\_mux \rceil + \lceil \frac{N_c}{N} \rceil \times (1 + \lceil \log_2 op\_mux \rceil)\right)$$

where reg_mux and op_mux are calculated as described in Figure 16.

The average aggregate wire delay introduced into the clock cycle is estimated as the delay through an average-length wire with average fanout multiplied by the average number of wires in sequence between the clock boundaries.

$$wd.ml = d(avg\_wl, avgf) \times \left(1 + \lceil \log_2 \frac{reg\_mux}{\tau} \rceil + \frac{N_c}{N} \times \left(1 + \lceil \log_2 \frac{op\_mux}{\tau} \rceil\right)\right)$$

The lower-bound wire delays introduced into the clock cycle is estimated as the delay through a minimum-length with wire minimum fanout, multiplied by the minimum number of wires in sequence between two clock boundaries (minimum 2 wires; one from a register to an operator and one from the operator to a register).

$$wd.lb = 2 \times d(0, 1)$$

A.7 Complexity

The run-time complexity of BEST is $O(n^2m)$ where n is the number of operations in the behavioral specification and m is number of library configurations. BEST tries $m = \Pi_i m_i$ library configurations where i varies over the operation types and m_i is the number of library operators of type i. This run-time complexity is for the generation of $O(nm)$ estimation points, resulting in $O(n)$ complexity per potential design explored. The run-time of BEST averages about 0.5 msec of CPU time per estimated design on a Sun Sparc 4/460 (180 msecs for 346 estimated designs).

A.8 Experimental Results

Two groups of validation efforts will be reported here. The experiments were performed with technology-dependent estimation techniques (PLA, wire delays, routing area, etc.) tuned for 3-micron and 1.2-micron CMOS technology, respectively. This section reports sample results from these experiments. In both sets of experiments, the ADAM Synthesis System was used to produce the RTL designs. Some schedules were modified manually when synthesis tools missed obvious choices to obtain results closer to what we would expect from the current generation of synthesis tools, but not all synthesized designs were screened for better optimization opportunities.[9] For experiments with the 3-micron technology, PLAs were generated using ESPRESSO and MKPLA, and for the 1.2 micron technology the Cascade Chipcrafter tools were used [25]. The layouts for synthesized designs were produced by the Chipcrafter System. In both set of experiments, we have not tried to generate a layout/design corresponding to each estimated design from BEST due to the amount of work needed.

The pre-layout (RTL + PLA) comparisons are shown in Figures 17 and 18 for non-pipelined implementations of FIR [20] and Elliptic Wave Filters [11]. Only a single library configuration (add3, mul2, register, mux) from the 3-micron library in Table 1 was used in this experiment.

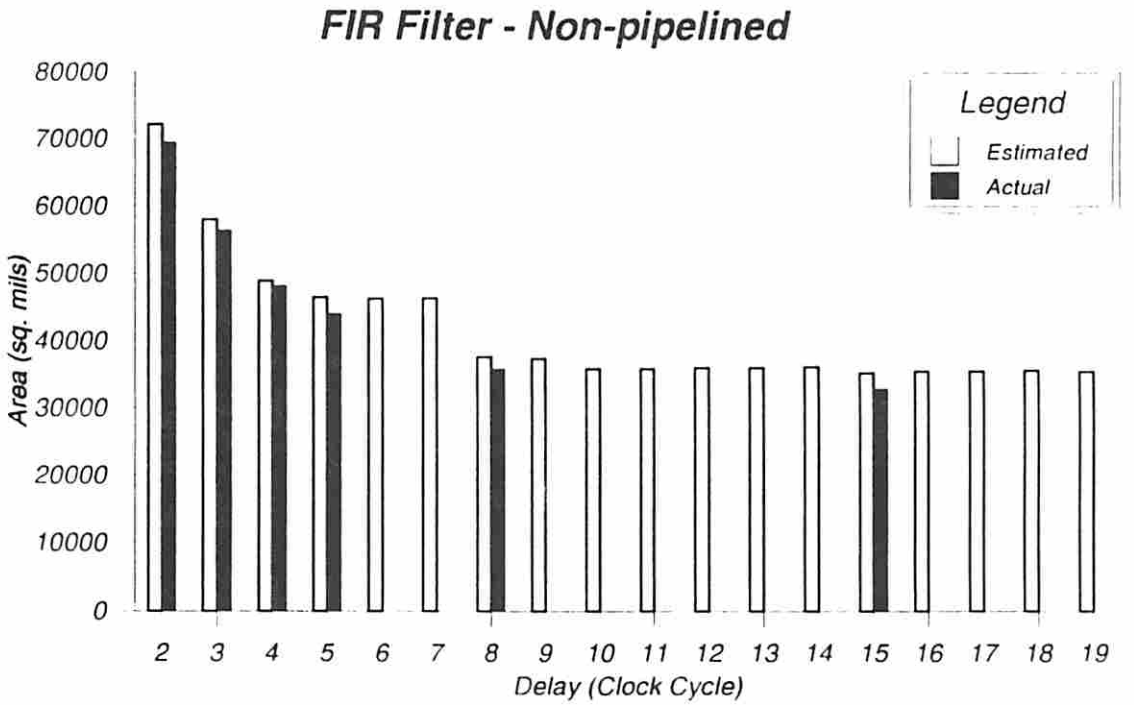

Figure 17: RTL + PLA area for the non-pipelined FIR Filter (3-micron).

Post-layout comparisons for the AR Filter Element from [6] are shown in Figure 19. It is important to note that there are differences between the layout design style assumed by BEST to perform estimations and the design style of the actual layouts. Layout estimations

[9]The fact that the predictions were able to point out inadequacies in the schedules used demonstrates another value of prediction. It also demonstrates that the predictions are not simple heuristics matched to one's own set of tools.

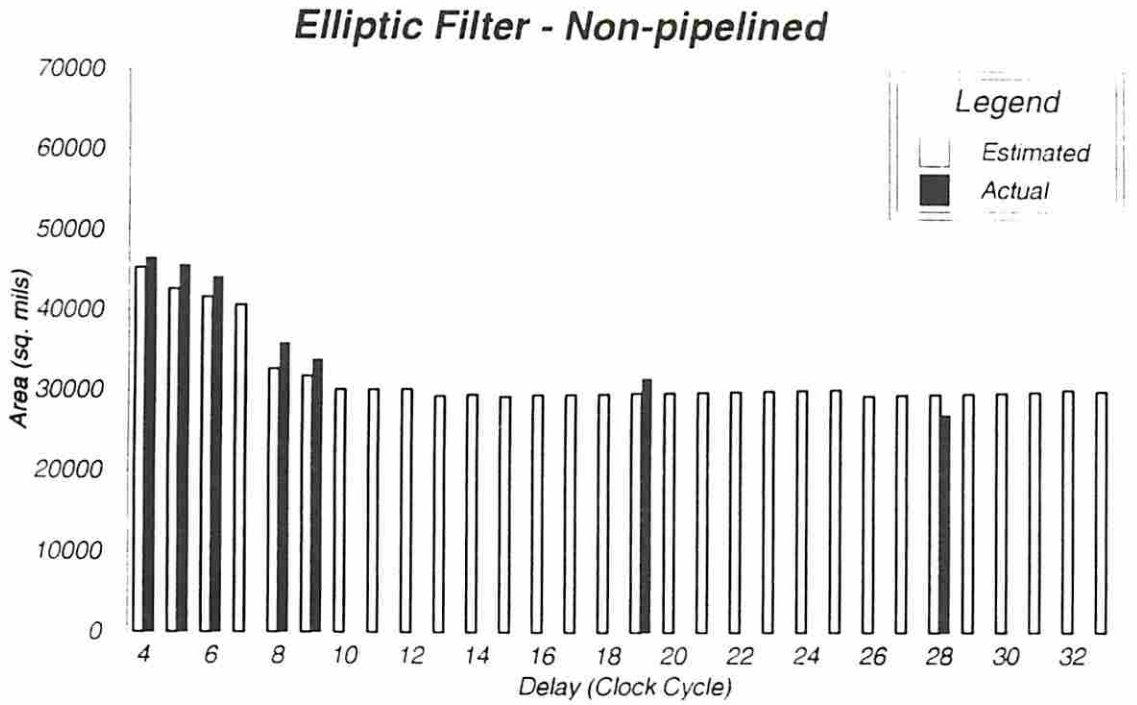

Figure 18: RTL + PLA area for the non-pipelined Elliptic Filter (3-micron).

in BEST assume a standard-cell implementation in contrast to the custom block implementation used by the Chipcrafter System.[10] BEST assumes that only 2-to-1 multiplexers are used while larger size multiplexers (3-to-1, 4-to-1, 10-to-1, etc.) which were better optimized for area and speed were used in the layouts. These differences make us believe that good layouts from the Chipcrafter System would generally be smaller and faster than the estimated counterparts.

Two more layouts for the pipelined design style were generated. The layout area (clock-cycle time) differences between the synthesized and estimated pipelined designs were 20%(8%) and 13%(4%) respectively. The clock-cycle time of the layouts were measured by the Chipcrafter tools.

[10]Chipcrafter was the most appropriate tool available to us to produce layouts. Other tools available at that time assumed design styles different from (and more rigid) than ADAM's.

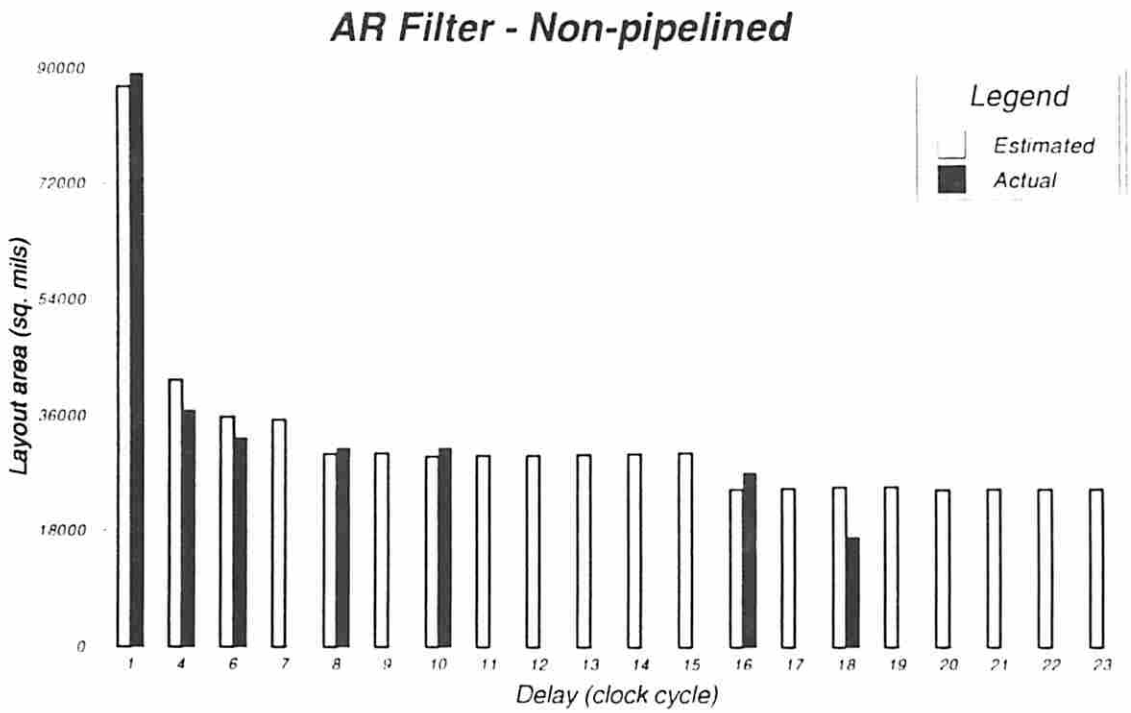

Figure 19: Layout area for the non-pipelined AR Filter (1.2-micron).