# Flexible ASIC: Shared Masking for Multiple Media Processors

Jennifer L. Wong
Univ. of Calif., Los Angeles
Los Angeles, California
jwong@cs.ucla.edu

Farinaz Kourshanfar
Univ. of Calif., Berkeley
Berkeley, California
farinaz@eecs.berkeley.edu

Miodrag Potkonjak
Univ. of Calif., Los Angeles
Los Angeles, California
miodrag@cs.ucla.edu

## ABSTRACT

ASIC provides more than an order of magnitude advantage in terms of density, speed, and power requirement per gate. However, economic (cost of masks) and technological (deep micron manufacturability) trends favor FPGA as an implementation platform. In order to combine the advantages of both platforms and alleviate their disadvantages, recently a number of approaches, such as structured ASIC/regular fabrics, have been proposed. Our goal is to introduce an approach that has the same objective, but is orthogonal to those already proposed. The idea is to implement several ASIC designs in such a way that they share the datapath, memory structure, and several bottom layers of interconnect, while each design has only a few unique metal layers. We identified and addressed two main problems in our quest to develop a CAD flow for realization of such designs. They are: (i) the creation of the datapath, and (ii) the identification of common and unique interconnects for each design. Both problems are solved optimally using ILP formulations. We assembled a design flow platform using two new programs and the Trimaran and Shade tools. We quantitatively analyzed the advantages and disadvantages of the approach using the Mediabench benchmark suite.

**Categories and Subject Descriptors:** B.5.1 [Design Aids]: Optimization

**General Terms:** Design.

**Keywords:** ASIC, interconnect, optimization.

## 1. INTRODUCTION

The traditional dilemma for many design teams has been ASIC or FPGA. FPGAs provide significant advantages over ASIC in terms of turn-around-time, flexibility, testability and suitability for debugging. However, for a given technology ASIC provides over 12 times higher speed, almost 50 times higher density, and more than 500 times lower power per gate [18]. Exponential growth of mask cost and a more demanding manufacturability process with each new generation of deep submicron technology caused two major ramifications on implementation platforms. Specifically, while in 0.25 micron the Non-Recurring Engineering (NRE) cost (mask set and probe card) was $100,000, it will approach $2 million for 65 nm [18]. At the same time printability and process variation challenges favor regular designs [13]. The first consequence is that regardless of great delay, area, and ASIC advantages, the number of ASIC designs has to be consistently decreasing. For example, over the last 15 years, the number of ASIC designs has been reduced 6 times (from 15,000 to 2,500). ASIC is currently used only for very large volume designs. If we consider a relatively large half of million volume, in 65 nm, the NRE cost will add $4 per design, which is often unacceptable overhead. The second consequence is the emergence of a quest for new design implementation platforms and design methodologies to develop solutions that preserve some of the advantages of ASIC while reducing NRE costs and increasing manufacturability.

A number of implementation platforms aimed to reduce mask and therefore NRE costs and to bridge the gap between ASICs and FPGAs have recently been proposed. The new industrial platform is commonly called structured ASIC or hybrid-FPGA structures and is pursued by large semiconductor companies, CAD houses, and start-ups [4, 5, 7, 9, 12, 15, 17, 18]. Academic researchers often refer to this implementation platform as regular fabrics and have focused their efforts mainly on either defining the structure of the fabric or exploring how the physical design is impacted by the emerging implementation platforms [2, 6, 7, 8, 11, 13, 14]. Most often the new platform consists of an array of logic cells built using diffusion and a few bottom metal layers. Logic cells contain programmable combinatorial logic and often storage elements. Therefore, the bottom mask layers are common to all designs that will be executed on this platform. Note that in addition to logic cells that can be customized, the interconnect network on the top few layers is generated using custom masks specifically crafted for a specific design. Therefore the approach significantly reduces the number of required custom masks and preserves some of the advantages in terms of delay, power, and area of a standard cell-based ASIC implementation platform. It is estimated that performances of this platform are inferior to fully custom ASIC by a factor of 2-4. However, specialized structure ASIC CAD tools are required.

Although our approach explores exactly the same division of layers in terms of common and customized, the approach is complementary to the ones already presented. Note that a few bottom layers in our case can be either fully customized


*DAC 2005,* June 13–17, 2005, Anaheim, California, USA.

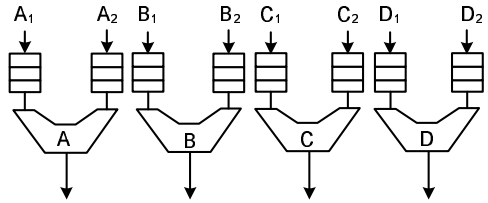


**Figure 1: Basic architecture of media applications.**

and implemented using cell-based ASIC, or can leverage on the bottom few layers of a structured ASIC. In the former case, we preserve all ASIC advantages in terms of area, delay, and power. In the later case, the emphasis is on enhanced manufacturability through regularity and reduced testing and debug costs. Furthermore, we focus on the register transfer and behavioral levels for the development of CAD tools for the effective use of new ASIC implementation platforms with reduced cost and shared bottom layers.

We identified two main aspects for developing CAD tools for flexible top layer ASIC implementation platforms. The first problem is how to select functional units that will address the needs of all applications. The second problem is how to assign interconnect to both the shared and customized metal layers. We proved that both problems are NP-complete and can be optimally solved using integer linear programming (ILP) formulations. Since the size of the instances for realistic designs are relatively moderate, the runtime of the CPLEX ILP solver in all cases was less than 0.1 second.

## 1.1 Motivational Example

The key idea behind the flexible ASIC approach is to generate interconnect assignments for different applications in such a way that all the applications have the same basic mask layers. The only difference would be the interconnect configurations for the upper layers. The approach can be introduced and illustrated using the following example. Assume that we have five different application-specific media processors, namely JPEG encoder, MPEG encoder, GSM encoder, G.721 encoder and Pegwit encoder. For the sake of clarity and simplicity, we show a very simplistic view of these encoder architectures. Assume that the basic architecture shown in Figure 1 should be used for building all the encoders and only the interconnect requirements differ. We have four different functional units denoted by $A$, $B$, $C$, and $D$. Each unit has an output with the same name as its block. There are two inputs for each blocks denoted as $A_1$, $A_2$, $B_1$, $B_2$, $C_1$, $C_2$, $D_1$, and $D_2$ respectively.

Table 1, shows the interconnect requirements for these five different media processors. Each row of the table represents an interconnect between the output of a block to the input of another block. Each column of the table represents a design. Assume that the chip has nine layers and requires that each layer can only have a maximum of two interconnects. If we directly implement each design, we will need five ASICs each with nine layers of interconnect, for a total of 45 interconnect masks.

However, by using the concept of flexible ASIC and analyzing the common interconnects required for all designs, the interconnect requirements can be organized as in Table 2. The leftmost column of Table 2 shows the inputs to each block. Each column afterward shows the output that

| | Application | | | | |
|---|---|---|---|---|---|
| IC | JPEG | MPEG | GSM | P.721 | Pegwit |
| A$A_1$ | • | • | • | • | • |
| A$A_2$ | - | - | - | • | • |
| A$B_1$ | • | • | - | - | • |
| A$B_2$ | • | • | • | • | • |
| A$C_1$ | • | • | • | • | • |
| A$D_2$ | - | • | • | - | - |
| B$A_2$ | • | • | • | • | • |
| B$B_1$ | • | • | • | • | • |
| B$C_1$ | • | • | • | - | - |
| B$C_2$ | • | • | • | • | • |
| B$D_1$ | - | - | - | • | • |
| B$D_2$ | - | - | • | • | - |
| C$A_1$ | • | • | - | - | • |
| C$B_1$ | • | • | • | • | • |
| C$B_2$ | • | • | • | • | • |
| C$D_1$ | • | • | • | • | • |
| D$A_1$ | • | • | • | • | • |
| D$B_2$ | • | • | • | • | • |
| D$C_1$ | • | • | • | • | • |
| D$C_2$ | - | - | • | • | - |
| D$D_1$ | • | • | • | • | • |
| D$D_2$ | • | • | • | • | • |

**Table 1: Interconnects required for each of the media processors.**

| Input | Basic | JPEG | MPEG | GSM | P.721 | Pegwit |
|---|---|---|---|---|---|---|
| $A_1$ | A,D | C | C | - | - | C |
| $A_2$ | B | - | - | - | A | A |
| $B_1$ | B,C | A | A | - | - | A |
| $B_2$ | A,C,D | - | - | - | - | - |
| $C_1$ | A,D | B | B | B | - | - |
| $C_2$ | B | - | - | B | B | - |
| $D_1$ | C,D | - | - | - | B | B |
| $D_2$ | D | - | A | A,B | B | - |

**Table 2: The input/output relationship for the encoder applications.**

enters the corresponding input. The second column shows the basic architecture interconnection that is the same for all five different applications. Columns 3 through 6 indicate the outputs per media application that are the required interconnects with each of the inputs on each row, in addition to the basic interconnects. Therefore, the first seven layers are completely dedicated to the 14 interconnects included in the basic architecture (column 2 of Table 2). The remaining interconnects are assigned to the remaining two layers in such a way that we have the minimum number of variants per layer and completely implement each of the media processors.

We find the following four variants (instantiation for the interconnects on one layer) for our five encoders: $v_1$: $CA_1$, $AB_1$, $v_2$: $CA_1$, $AB_1$, $v_3$: $BD_2$, $DC_2$, $v_4$: $BD_1$, $AA_2$.

By looking at the interconnect requirements from Table 2, we see that variants $v_1$ and $v_2$ can be used for realizing the MPEG and the JPEG decoders. Variants $v_2$ and $v_3$ can realize the GSM encoder. Variants $v_3$ and $v_4$ can realize the G.721 encoder, while variants $v_1$ and $v_4$ can realize the Pegwit encoder. Therefore, the addition of only two layers to the basic layers can realize the full architecture for all five encoders. For each architecture the first seven layers are identical in terms of masks, while the last two layers are realized by two of the four variants corresponding to each architecture and are hence different. Therefore, using flexible ASIC only eleven unique metal layers are necessary to realize all five media processors.

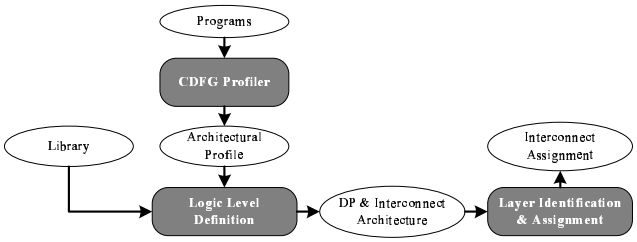


**Figure 2: Global Flow.**

In the following section we briefly discuss the related work. In Section 2 we identify the overall flow of the approach for realization of multiple applications on a single chip. We introduce our ILP approach for determining the basic layers of the chip in Section 3. We formulate the interconnect assignment problem using integer linear programming (ILP) for two optimizations: minimal variant layers and minimal design and manufacturing cost. Before concluding the work we present the experimental analysis of the technique.

## 2. DESIGN FLOW

Figure 2 shows the flow of our flexible ASIC approach. The input to the CAD system is a set of applications specified using C or VHDL programs. In our experimentation, we focused on the implementation of large media applications that have been specified and simulated for their functionality correctness using C programs. Specifically, we considered all Mediabench applications [10] and the Viterbi convolution encoder. All programs have been profiled for the identification of the number of operations and transfers of each type. Profiling is accomplished through a relatively simple wrapper around the Trimaran [16] and Shade tools [1]. The output of the CDFG profiler is the required datapath and interconnect requirements of each application specified at the behavioral level. This profile is an input, together with a library of the available functional units, to a program that identifies the best units for inclusion in the datapath in such a way the the likelihood that all timing constraints for all applications are satisfied and the area of the design minimized. Note that we do not specifically optimize the memory structure. In practice we observed that it is sufficient to allocate memory requirements of the most intensive application for all others.

The result of the datapath definition module has two components. The first is a set of multi-functional units that satisfy the specified clock cycle time. The second is the set of required interconnects for each application. The second component is the input into the layer identification and assignment program that minimizes the total number of unique layers required for all applications. We considered two variants of the layer identification assignment problem: (i) where the goal is to solely minimize the number of unique layers, and (ii) where the layers are identified and assigned in such a way that the sum of the NRE and manufacturing cost is minimized. Both the datapath and layer identification assignment programs are solved using ILP. For this purpose we used the CPLEX solver.

## 3. DATAPATH DEFINITION

In this section, we first formulate the datapath definition problem and its complexity. We present the ILP formulation of the problem which can be solved using any ILP solver.

We assume that a set of applications is given. The goal is to allocate a datapath for these applications using the available library of functional units. The library contains specifications for a number of unit types. The goal is to find a mapping which minimizes the total area used by the logical level definition (actual functional units) of our applications. The problem can be more formally stated as follows. Note that, we define the problem instance in such a way that we can directly use it as the input to our ILP formulation in the next subsection.

---

**Problem:** *Datapath Definition Problem*
**Instance:**
• *A set of applications* $MP_k$, $k = 1, ..., N_k$.
• *A number of operations* $OP_j$, $j = 1, ..., N_j$.
• *A set of required operations* $RO_{kj}$, *that define if the operation* $OP_j$ *is required by the application* $MP_k$, $j = 1, ..., N_j$, $k = 1, ..., N_k$:

$$RO_{kj} = \begin{cases} 1, & \text{if operation } OP_j \text{ is required by } MP_k, \\ 0, & \text{otherwise.} \end{cases}$$

• *The architectural profiles* $P_{kj}$, *that defines the number of operations of type* $OP_j$ *in the applications* $MP_k$, $j = 1, ..., N_j$, $k = 1, ..., N_k$.
• *A number of for functional unit types (families)* $TFU_i$, $i = 1, 2, ...., N_i$.
• *A library with elements* $L_{ij}$ *such that each type of functional units* $TFU_i$ *can perform the operations* $OP_j$, $i = 1, 2, ...., N_i$, $j = 1, ..., N_j$:

$$L_{ij} = \begin{cases} 1, & \text{if } TFU_i \text{ can perform operations of type } OP_j, \\ 0, & \text{otherwise.} \end{cases}$$

• *An actual set of functional units* $FU_a$, $a = 1, ..., N_a$;
• *An area cost* $A_i$, *for each functional unit type* $TFU_i$, $i = 1, 2, ...., N_i$.
• *A throughput constraint* $T_k$ *corresponding to each application* $MP_k$, $k = 1, ..., N_k$.
**Question:** *Is there a subset of functional units* $FU_a$, $a = 1, ..., N_a$, *such that the subset has the cardinality* $N_{sub}$ *and all applications are realizable (functional units and transfers) and the total cost is minimized?*

---

### 3.1 NP-Completeness

The datapath definition problem is NP-complete. It is a special case of the minimum cover problem. The minimum cover problem is formulated as following in Garey-Johnson format [3].

---

**Problem:** *[SP5] Minimum cover*
**Instance:** *Collection* $C$ *of subsets of a finite set* $S$, *positive integer* $K \leq |C|$.
**Question:** *Does* $C$ *contain a cover for* $S$ *of size* $K$ *or less, i.e., a subset* $C' \subseteq C$ *with* $|C'| \leq K$ *such that every element of* $S$ *belongs to at least one member of* $C'$*?*

---

We prove that the minimum cover problem is a special case of the datapath definition problem. Each element in the set cover problem corresponds to a type of operation in the specification of the designs. Each collection corresponds to a functional unit in the library that can execute a set of operations that correspond to the elements of the set cover problem. The goal is to select at most $K$ collections, or

functional units, which implement all the operations. Note that the throughput constraint on the functional units is not enforced in the case of the minimum cover problem.

## 3.2 ILP Formulation of the Problem

In order to solve this problem using ILP, a number of variables are defined. The first set of variables are denoted by $x_{ai}$ and relate the actual functional units $FU_a$ to the type of functional units $TFU_i$, $a = 1, ..., N_a$, $i = 1, 2, ...., N_i$:

$$x_{ai} = \begin{cases} 1, & \text{if functional units } FU_a \text{ is of type } TFU_i, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The second set of variables are denoted by $t_{kja}$ and are defined for each application $MP_k$, and determine the number of operations of type $OP_j$ assigned to the functional unit $FU_a$, , $k = 1, ..., N_k$, $j = 1, ..., N_j$, $a = 1, ..., N_a$.

The objective function is to minimize the total cost of the implementation area required for implementation of the applications (actual functional units):

$$min \qquad \sum_i \sum_a A_i x_{ai}$$

We introduce four sets of constraints for the problem. The first set of constraints ($C_1$), ensures that each functional unit $FU_a$ is only of one specific type $TFU_i$. The second set of constraints ($C_2$), enforces each application to be realizable by the selected functional units. Thus, for each application $MP_k$, all operations of type $OP_j$, must be implemented by at least one selected functional unit $FU_a$:

$$C_1: \quad \forall a, \quad \sum_i x_{ai} = 1$$

$$C_2: \quad \forall k, \forall j, \quad \sum_a \sum_i O_{kj} L_{ij} x_{ai} \geq 1$$

Not only must each of the applications be realizable, but the throughput constraint for each application should be met. The third and fourth sets of constraints enforce the throughput constraints for each application and each type of operation. The third set of constraints ($C_3$) ensures that all instances of the operation are assigned to a selected functional unit which can perform the operation. $C_3$ consists of two notions, the first of which specifies if the selected functional unit $FU_a$ of type $TFU_i$ can perform the operation $OP_j$. If so, then this functional unit can be assigned the operation of type $OP_j$. Secondly, the sum of all operations of type $TFU_j$ assigned to all selected functional units must equal the total number of operations required for $MP_k$. We implement these requirements using the logical AND ($\wedge$) function. AND is implemented in terms of ILP constraints by introducing a new variable and adding three constraints for each AND operation. Specifically, $a \wedge b = c$ is implemented in the following way where $c$ is a binary variable.

$$a + b - c \leq 1 \quad a - c \geq 0 \quad b - c \geq 0$$

The fourth and last set of constraints ($C_4$), ensures that all the assigned operations to an actual functional unit for a given application do not exceed the throughput limitations of that application. For each application $MP_k$, and for each selected functional unit $FU_a$, for any type of operation $OP_j$ that the functional unit of a certain type can perform, the sum must be less than the throughput specified for the operation:

$$C_3: \quad \forall O_{kj} = 1, \quad \sum_a \sum_i (L_{ij} x_{ai} \wedge t_{kja}) = P_{kj}$$

$$C_4: \quad \forall k, a, \quad \sum_j \sum_i (L_{ij} x_{ai} \wedge t_{kja}) \leq T_k$$

# 4. INTERCONNECT ASSIGNMENT

Once the datapath for each of the processors is defined (i.e. the solution from the datapath definition problem), the next step is to determine the interconnect assignments for each of the processors. In this section, we first formulate the interconnect assignment problem and the complexity of the problem. Next, we present the ILP formulation of the problem. Finally, we introduce a variant on the original problem formulation which addresses the minimization of the total implementation cost of all layers.

Assume the interconnect specifications for a number of application-specific media processors $MP$s is known. In addition, the specific interconnects for each of the media processors are given. The fabrication process has a constraint on the number of interconnects per layer. The goal is to assign interconnects to the different layers for each application in such a way that the maximum number of layers can be shared between the applications, only a few layers vary for different applications, and the constraints on the number of interconnects on one layer are satisfied. More formally the problem can defined as follows.

---

**Problem:** *Interconnect Assignment Problem*
**Instance:** *Application-specific media processors ($MP_k$) with specified interconnects, number of layers ($S$), number of interconnects per layer ($M$), number of allowable variations for each layer ($R$).*
**Question:** *Is there an assignment of interconnects to each layer s.t. the total number of created variations are for all layers are minimal, and each $MP_k$ is created?*

---

## 4.1 NP-Completeness

The interconnect assignment problem is NP-complete. It is a special case of the Largest Common Subgraph problem. The Largest Common Subgraph problem is formulated in the Garey-Johnson format [3] as:

---

**Problem:** *[GT49] Largest Common Subgraph*
**Instance:** *Graph $G = (V_1, E_1)$, $H = (V_2, E_2)$, positive integer $K$.*
**Question:** *Do there exist subsets $E_1' \subseteq E_1$ and $E_2' \subseteq E_2$ with $|E_1'| = |E_2'| \geq K$ such that the two subgraphs $G' = (V_1, E_1')$ and $H' = (V_2, E_2')$ are isomorphic?*

---

Note that the interconnect requirements for each application form a graph, where the vertices of the graphs are the functional blocks of the MPs and the edges are the interconnects between the blocks. When we have only two applications, the selected subset of edges is the set of interconnects which are shared by the media processors. In this case the goal is to find the maximal set.

## 4.2 ILP Formulation

We now show the notations and the variables that are used for formulating the interconnect assignment problem in

ILP. We assume that the total number of layers is specified $s = \{1, \ldots, S\}$, as well as the maximum number of variants per layer $r = \{1, \ldots, R\}$. The maximum number of interconnects on one layer is denoted by a constant $M$ and is an input to the problem. The three dimensional matrix $W_{ijk}$ is an input to the problem and indicates the data flow edges (interconnects) for each application $k$. More specifically,

$$W_{ijk} = \begin{cases} 1, & \text{if the edge (i,j) exist in application k,} \\ 0, & \text{otherwise.} \end{cases}$$

The solution to the problem finds different variants (instantiations) for each of the layers. The four dimensional variable $x_{ijrs}$ is used to denote the interconnect assignment for each layer in each of the variants. More formally,

$$x_{ijrs} = \begin{cases} 1, & \text{if the edge (i,j) exist in variant r on layer s,} \\ 0, & \text{otherwise.} \end{cases}$$

Additionally, we define a three dimensional variable, $u_{krs}$, that links each application to the layers and the variant on that layer on which its interconnect requirements are implemented.

$$u_{krs} = \begin{cases} 1, & \text{if the application k uses variant r on layer s,} \\ 0, & \text{otherwise.} \end{cases}$$

Lastly, we define a variable to denote if a possible variant is used on each layer. The two dimensional variable $y_{rs}$ is such a variable and is defined below.

$$y_{rs} = \begin{cases} 1, & \text{if variant r is used on layer s,} \\ 0, & \text{otherwise.} \end{cases}$$

Note that the variables $x_{ijrs}$, $u_{krs}$, and $y_{rs}$ are not independent of each other and we will define a set of constraints in the problem formulation that relates these three variables.

The objective function of the problem is to minimize the total number of variants on all of the chip layers. This objective function can be formally written as:

$$min \qquad \sum_r \sum_s y_{rs}$$

There are five constraints to the formulation which relate the above variables together and specify the limitations on the variables. The first constraint of the problem ($C_5$) ensures that each of the required interconnects for all the applications are implemented on at least one variant on any of the layers of the chip. This ensure that each of the media applications can be realized on the chip. More formally,

$$C_5: \quad \forall k \in [1, K], \forall \text{ (i,j)} \qquad W_{ijk} \leq \sum_r \sum_s x_{ijrs}$$

$$C_6: \quad \forall s, \forall r, \qquad \sum_i \sum_j x_{ijrs} \leq M$$

$$C_7: \quad \forall k \in [1, K], \forall \text{ layers } s, \quad \sum_r u_{krs} \leq 1$$

The second constraint of the problem ($C_6$) is to ensure that each layer has at most the maximum number of $M$ interconnects at each layer, i.e., When realizing each of the applications the required interconnects must all be implemented on only one variant per layer. To ensure this, the third constraint ($C_7$) specifies that each application can only be assigned at most one variant at each layer. In other words, The two final constraints ($C_8$ and $C_9$) relate all the variables to each other. In particular, the constraint $C_8$ determines the relationship between the overall used layers, $y_{rs}$, and the layers used by each specific application, $u_{krs}$. The relationship is straightforward from the definition of the two variables. If an application $k$ uses variant $r$ on layer $s$ (ie. $u_{krs} = 1$), then the variant $r$ on layer $s$ must be used. This relationship is shown as follows.

$$C_8: \quad \forall k \in [1, K] \quad y_{rs} \geq u_{krs}$$

$$C_9: \quad \forall i, \forall j \qquad u_{krs} \geq W_{ijk} x_{ijrs}$$

Constraint $C_9$ specifies the relationship between the interconnects on each variant of each layer, $x_{ijrs}$, and the variant layers which are used for each application, $u_{krs}$. This relationship is also directly concluded from the definitions. If the edge (i,j) exists in variant $r$ on layer $s$, and also the edge (i,j) is required for realization of application $k$, then $x_{ijrs}$ and $W_{ijk}$ are one. In such a situation, the application $k$ can use the variant $r$ on layer $s$, i.e. $u_{krs} = 1$.

## 4.3 Interconnect Assignment for Cost

In this Subsection the focus of the optimization switches from minimizing the number of total variant layers to minimizing the monetary cost for implementation of the layers, both design and manufacturing.

We assume that the cost of designing each of the layers is known along with the amount of each media processor to be manufactured. Considering these new inputs the problem can be formally defined as follows.

---

**Problem:** *Cost Minimization with Interconnect Assignment*
**Instance:** *Application-specific media processors ($MP_k$) with specified interconnects, number of layers ($S$), number of interconnects per layer ($M$), number of allowable variations for each layer ($R$), cost for design for each layer ($D$), volume of manufacturing each media processors ($V_k$).*
**Question:** *Is there an assignment of interconnects to each layer s.t. the total cost of design and manufacturing all designs is minimal and each $MP_k$ is created?*

---

In order to address this problem which is an extension of the interconnect assignment problem, we reformulate the objective function to focus on the total cost rather than the number of layers. The total cost is the cost to design each of the necessary layers, plus for each layer the volume of manufacturing necessary for each of the media processors. Therefore, using the same formulation variables from the interconnect assignment problem we formulate the objective function as:

$$min \qquad D * (\sum_r \sum_s y_{rs}) + \sum_k (V_k * (\sum_r \sum_s u_{krs}))$$

All constraints of the problem are identical to the interconnect assignment problem.

# 5. EXPERIMENTAL RESULTS

Flexible ASIC is particularly well suited for computational intensive applications such as media processors. Therefore, for evaluation purposes we used 21 applications from the Mediabench benchmark suite [10] and the Viterbi decoder that is widely used in both wireless and optical communications. The Viterbi decoder was obtained from the LSI Logic corporation and is denoted in Table 3 by V. The Mediabench

| designs | # of designs | # of initial layers | # of custom layers | # of final layers | reduction | initial cost ($M) | final cost ($M) | cost reduct |
|---|---|---|---|---|---|---|---|---|
| A,B,C,D,V | 5 | 49 | 2 | 14 | 71.43% | 9.9 | 6.4 | 35.35% |
| G,H,I,J,V | 5 | 49 | 2 | 15 | 69.39% | 9.9 | 6.5 | 34.34% |
| A,B,L,M,N | 5 | 47 | 2 | 16 | 65.96% | 9.7 | 6.6 | 31.96% |
| A,B,C,D,I,J | 6 | 60 | 2 | 17 | 71.67% | 11 | 7.7 | 30.00% |
| G,H,I,J,M,N | 6 | 58 | 2 | 17 | 70.69% | 10.8 | 7.7 | 28.70% |
| O,P,Q,R,S,T | 6 | 52 | 3 | 15 | 71.15% | 10.2 | 7.5 | 26.47% |
| A,B,C,D,G,H,I,J | 8 | 80 | 2 | 16 | 80.00% | 16 | 9.6 | 40.00% |
| A,B,E,F,G,H,I,J | 8 | 80 | 3 | 15 | 81.25% | 16 | 9.5 | 40.63% |
| A,B,G,H,I,J,L,M,N,V | 10 | 96 | 3 | 20 | 79.17% | 19.6 | 12 | 38.78% |
| A,B,C,D,G,H,I,J,K,L | 10 | 98 | 3 | 18 | 81.63% | 19.8 | 11.8 | 40.40% |

**Table 3: Analysis of the new flexible ASCI scheme in terms of technological complexity (number of layers) and estimated cost for the case where on manufacturing of each design $1M is spent per design.**

designs have been denoted using the following notation : Cjpeg (A), Djpeg (B), MPEG2encode (C), MPEG2decode (D), toast (E), untoast (F), rawcaudio (G), rawdaudio (H), G.721 encode (I), G.721 decode (J), PGPencode (K), PGPdecode (L), PEGWITencode (M), PEGWITdecode (N), Ghostscript (O), mipmap (P), osdemo (Q), texgen (R), rasta (S), EPIC (T), and unEPIC (U). We organized all the designs into ten groups by similarities of their applications and similarity in their type of datapath and interconnect requirements. The smallest group consisted of five designs and the largest contained ten designs. For all designs we assumed the total number of available layers was ten and that each metal layer can be used to implement at most five global interconnect between different functional units. In all examples, the datapath consisted of eight units.

Table 3 shows characteristics of the designs when implemented using the traditional ASIC where each design is implemented separately and after application of the new flexible ASIC design methodology and platform. The first column indicates which designs were considered for simultaneous implementation on a single platform using the flexible ASIC methodology. The second column shows the total number of layers required for all design when the traditional ASIC approach is used. The fourth column indicates the number of customized layers used by the flexible ASIC approach. The next two columns show the total number of layers required by the flexible ASIC approach and the total reduction in percentage of layers required for implementation. The last three columns are used to estimate the economic advantage of flexible ASIC under the assumption that the cost of a layer is $100,000 and that the manufacturing cost for each type of design in $1M. Note that the reduction in number of layers is equivalent to the reduction in NRE cost. The average NRE cost was reduced by 74% and the average overall cost reduction by 34%. Note that in all cases, the cost reduction increases as the number of designs targeted to share the common bottom layers increases.

# 6. CONCLUSION

In order to reduce NRE cost and enhance the applicability of ASIC implementation platforms, we introduced a common bottom - customized top layers flexible ASIC approach. The approach provides a new alternative between FPGA and ASIC implementation platforms: it essentially maintains ASIC advantages in terms of gate speed, density, and power, while reducing the NRE cost. We identified the two key phases for CAD tools that target the new ASIC platform and solved associated optimization problems optimally using the CPLEX ILP solver. The experimentation indicates that the new platform can be used both to reduce NRE and overall implementation costs.

# 7. REFERENCES


[1] R. F. Cmelik and D. Keppel. Shade: A fast instruction-set simulator for execution profiling. Technical Report SMLI 93-12, UWCSE 93-06-06, 1993.

[2] J. Cong, Y. Fan, X. Yang, and Z. Zhang. Architecture and synthesis for multi-cycle communication. In *International Symposium on Physical Design*, pages 190–196, 2003.

[3] M. Garey and D. Johnson. *Computers and Intractability.* W.H. Freeman, 1979.

[4] http://www.fma.fujitsu.com/accel/main01.asp.

[5] http://www.lsilogic.com/products/rapidchip_platform_asic/index.html.

[6] B. Hu, H. Jiang, Q. Liu, and M. Marek-Sadowska. Synthesis and placement flow for gain-based programmable regular fabrics. In *International Symposium on Physical Design*, pages 197–203, 2003.

[7] A. Kahng, I. Bolsens, J. Cohn, B. Gupta, C. Hamlin, Z. Orbach, and L. Pileggi. What is the next implementation fabric? *IEEE Design and Test of computers*, 20(6):86–95, Nov. 2003.

[8] V. Kheterpal, A. J. Strojwas, and L. Pileggi. Routing architecture exploration for regular fabrics. In *Conference on Design Automation*, pages 204–207, 2004.

[9] D. E. Lackey, P. S. Zuchowski, and J. Koehl. Designing mega-ASICs in nanogate technologies. In *Conference on Design Automation*, pages 770–775, 2003.

[10] C. Lee, W. Mangione-Smith, and M. Potkonjak. Mediabench: A tool for evaluating multimedia and communication systems. In *MICRO-30 Conference*, pages 330–335, Nov. 1997.

[11] F. Mo and R. K. Brayton. Fishbone: a block-level placement and routing scheme. In *International Symposium on Physical Design*, pages 204–209, 2003.

[12] T. Okamoto, T. Kimoto, and N. Maeda. Design methodology and tools for NEC electronics' structured ASIC ISSP. In *International Symposium on Physical Design*, pages 90–96, 2004.

[13] L. Pileggi, et al. Exploring regular fabrics to optimize the performance-cost trade-off. In *Conference on Design Automation*, pages 782–787, 2003.

[14] Y. Ran and M. Marek-Sadowska. On designing via-configurable cell blocks for regular fabrics. In *Conference on Design Automation*, pages 198–203, 2004.

[15] D. D. Sherlekar. Design considerations for regular fabrics. In *International Symposium on Physical Design*, pages 97–102, 2004.

[16] Trimaran. http://www.trimaran.org/.

[17] K. Wu and Y. Tsai. Structured ASIC, evolution or revolution? In *International Symposium on Physical Design*, pages 103–106, 2004.

[18] P. S. Zuchowski, C. B. Reynolds, R. J. Grupp, S. G. Davis, B. Cremen, and B. Troxel. A hybrid ASIC and FPGA architecture. In *International Conference on Computer-aided Design*, pages 187–194, 2002.