HEURISTIC MINIMIZATION FOR SYNCHRONOUS RELATIONS

by

Vigyan Singhal, Yosinori Watanaba, and
Robert K. Brayton

Memorandum No. UCB/ERL M93/30

29 April 1993

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

HEURISTIC MINIMIZATION FOR SYNCHRONOUS RELATIONS

by

Vigyan Singhal, Yosinori Watanabe, and
Robert K. Brayton

Memorandum No. UCB/ERL M93/30

29 April 1993

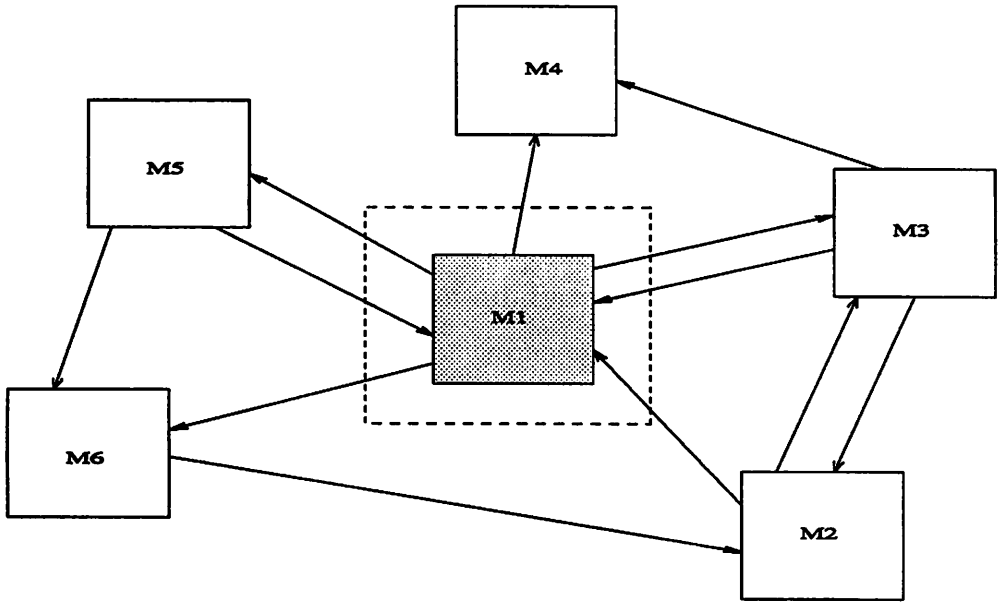


Figure 1: A system of interacting FSMs. The arrows indicate the communication between the FSMs. We wish to characterize the input-output behavior for the shaded machine $M1$.

minterms lie in the boolean space B^m and the output minterms lie in B^n. Synchronous relations, thus, can be used to express constraints among output patterns for different input minterms.

Synchronous relations arise naturally in many contexts in the design of synchronous digital systems. Consider a system of interacting finite state machines (FSMs) as in Figure 1. If we wish to optimize the implementation of one of the component machines $M1$, we can derive the don't care sequences for this component based on its environment (the rest of the FSMs). The input-output sequences will denote the specification for the component machine, and will represent some flexibility in its implementation. Such finite sequences can be expressed by synchronous relations. [RHS91] have showed how to obtain such finite behavior sequences for the special case of two cascaded FSMs— one driving machine and the other driven. At the end of Section 2.2 we will give an example of a situation where the don't care flexibility for the FSM behavior can be expressed by synchronous relations, but not by any ordinary boolean relation.

Instead of a network of FSMs as described above, we may wish to optimize a sequential design at the gate level comprised of combinational elements and latches. In such a situation, we wish to optimize a subcircuit based on the sequential implementation surrounding it.

Heuristic Minimization for Synchronous Relations

Vigyan Singhal, Yosinori Watanabe and Robert K. Brayton*
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720, USA

Abstract

Optimization for synchronous systems is an important problem in logic synthesis. However, the full utilization of don't care information for sequential synthesis is far from being solved. Synchronous boolean relations can represent sequential don't care information upto in synchronous systems. This allows greater flexibility in expressing don't care information than ordinary boolean relations relating input and output space. Synchronous relations can be used to specify sequential designs both at the finite state machine level as well as at the level of combinational elements and latches. In this report we also show that the synchronous relation formulation can also be used to find a minimal sum-of-products form which implements a function compatible with an arbitrary set of boolean relations. The main objective of this report is to present a heuristic approach to find a minimal implementation for a given synchronous relation.

1 Introduction

This report is concerned with the problem of finding minimal cost implementations for synchronous relations. Synchronous relation express the input-output behavior of synchronous systems. Using time-shifted variables it is possible to characterize sequences of input-output behavior using these relations. This allows the specification of don't care sequences for sequential designs; such a specification permits a strictly greater expressibility of don't care information than the ordinary boolean relations [BS89].

A synchronous relation S is a boolean relation which relates sets (of size d_1) of input minterms to sets (of size d_2) of output minterms, i.e. $S \subseteq B^{md_1} \times B^{nd_2}$, where the input

*This research was supported by NSF/DARPA Grant MIP-8719546, NSF Grant EMC-8419744 and by various grants from BNR, California Micro Program, DEC, Intel, AT&T and Motorola.

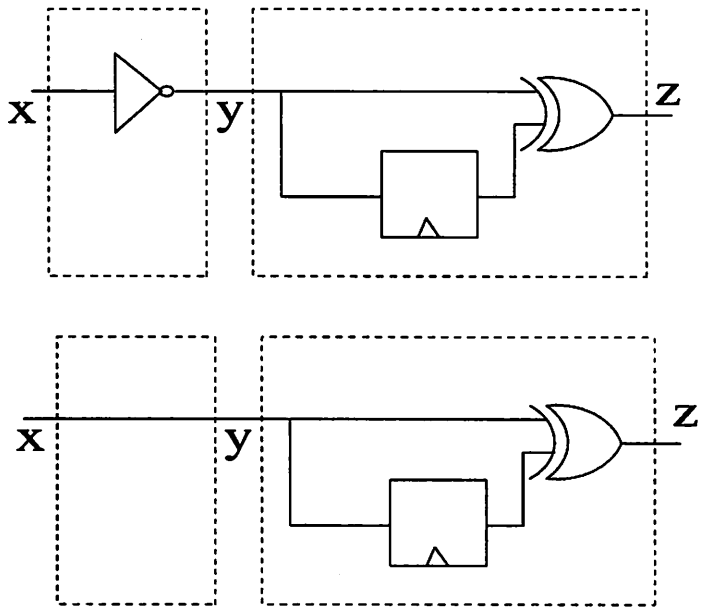


Figure 2: Optimization of combinational sub-circuit using synchronous relations

Damiani and De Micheli ([DDM92a, DDM92b]) have shown that synchronous relations can be derived to express the flexibility in such scenarios. They have also shown that synchronous relations are capable of expressing strictly greater flexibility than ordinary boolean relations. For an example of the use of synchronous relations at the gate-level design, consider the example circuit from [DDM92a] (the upper circuit in Figure 2). For minimizing the left half of the circuit so that the behavior of the total circuit is preserved, we can define a synchronous relation between $\mathbf{x}$ and $\mathbf{y}$. Using the time-shifted variables, this synchronous relation captures the observability don't cares introduced for the specification of the left half of the circuit. We can then minimize the left circuit using techniques described in this report. An alternate implementation of the left half which preserves the behavior of the total system is shown in the lower circuit. This cannot be obtained using Boolean relation minimization. While this example illustrates the minimization of a combinational circuit using sequential observability don't cares, we do not require that the minimization of synchronous relations be restricted to combinational implementations.

Besides minimization for sequential circuits, synchronous relations can be used for other applications where one needs to express constraints among sets of input-output minterms, for example [KF92]. The relationship between synchronous relations and specifications where such constraints occur has been explored in detail elsewhere ([SSB93]).

Once a synchronous relation S is given, the objective is to find a least cost implementation whose functionality satisfies S. If the resulting implementation is pure combinational logic, the implementation $f : B^m \rightarrow B^n$ satisfies synchronous relation $S \subseteq B^{md} \times B^{nd}$, said to be *compatible*, if and only if $\forall(\mathbf{x}^1, \ldots, \mathbf{x}^d) : S(\mathbf{x}^1, \ldots, \mathbf{x}^d, f(\mathbf{x}^1), \ldots, f(\mathbf{x}^d)) = 1$. The traditional approaches used to find functions compatible with ordinary boolean relations [BS89, WB91] cannot be used to find solutions for synchronous relations. This is because, unlike synchronous relations, compatibility of a function with an ordinary boolean relation cannot express constraints among outputs for different inputs[1].

Damiani and De Micheli ([DDM92a]) have given a procedure to explicitly compute the exact solution for a given synchronous relation. They have introduced additional variables to express the constraints between outputs. The number of additional variables is equal to the number of minterms in the input space (2^m). First they extract the prime implicant for a feasible compatible function. Then they express the synchronous constraints in terms of the additional variables, and solve a binate covering step, similar to the procedure in [BS89]. We believe with increase in depth d and number of input and output variables such an explicit and exact approach will become infeasible. We will use a heuristic approach and use implicit computation to solve the minimization problem.

In this report, we first consider the expressiveness of synchronous relations and show that a hierarchy of expressiveness exists in terms of the depth d. While there might exist a set of functions that cannot be represented by a given depth d, for any arbitrary set of functions, there exists another depth $d' > d$ which can represent the set. After that we present a heuristic approach to minimize a compatible representation for a synchronous relation.

Our heuristic approach is similar to the approaches used for minimization of incompletely specified functions (ESPRESSO) [BHMS84] or for minimization of boolean relations (GYOCRO) [WB91]. Specifically, we start with an initial representation in sum-of-products form and iteratively modify it, always maintaining compatibility with the synchronous relation. We always move in the direction of reducing the cost function and we manipulate only one cube in the representation at a time. The procedure represents and manipulates the synchronous relation and the compatible function implicitly using Binary Decision Diagrams (BDDs) [Bry86].

[1] A function $f : B^m \rightarrow B^n$ is compatible with boolean relation $R \subseteq B^m \times B^n$ if and only if $\forall \mathbf{X} : R(\mathbf{X}, f(\mathbf{X})) = 1$ [BS89].

$(\mathbf{x}^1, \mathbf{x}^2)$	$(\mathbf{y}^1, \mathbf{y}^2)$
$(0,0)$	$\{(0,0),(1,1)\}$
$(0,1)$	$\{(0,1),(1,0)\}$
$(1,0)$	$\{(1,0),(0,1)\}$
$(1,1)$	$\{(1,1),(0,0)\}$

Figure 3: Synchronous relation which expresses the don't care flexibility for Figure 2. For each value of $(\mathbf{x}^1, \mathbf{x}^2)$, the synchronous relation is satisfied if $(\mathbf{y}^1, \mathbf{y}^2)$ assumes any of the values in the corresponding set.

In the section 2 we introduce some preliminaries which are required to understand our formulation for the minimizer and present some theoretical results. We describe our minimization procedure in section 3 and present some results based on an implementation of the procedure described in this report.

2 Preliminaries

2.1 Terminology

In this section, we introduce some concepts which will be required for our formulation of the minimization of the implementation of synchronous relations.

Notation 1 *The* **consensus operator** *applied to a function f with respect to a variable x is denoted by $C_x f$, and is computed as $C_x f = f_x \cdot f_{\overline{x}}$. The* **smoothing operator** *S_x is computed as $S_x f = f_x + f_{\overline{x}}$.*

Definition 1 *A* **synchronous relation** *S of depth d is a subset of $B^{md} \times B^{nd}$. Let the input variable be denoted by $\mathbf{x} = (x_1, \ldots, x_m)$ and the output variables by $\mathbf{y} = (y_1, \ldots, y_n)$. The temporal values of the input and output variables at $j-1$ clock cycles before the current clock cycle are denoted by $\mathbf{x}^j = (x_1^j, \ldots, x_m^j)$ and $\mathbf{y}^j = (y_1^j, \ldots, y_n^j)$, respectively, where j varies between 1 and d.*

Example 2.1 *For the sequential circuit shown in upper figure in Figure 2, we can use a synchronous relation $S(\mathbf{x}^1, \mathbf{x}^2, \mathbf{y}^1, \mathbf{y}^2)$ to express all the flexibility we have in implementing the first sub-circuit with input $\mathbf{x}$ and output $\mathbf{y}$ such that the behavior of $\mathbf{z}$ in terms of $\mathbf{x}$ remains unchanged. This synchronous relation is shown in Figure 3.*

$\mathbf{x} \in B^1$	$\mathbf{y} \in B^1$	
	f_1	f_2
0	0	1
1	1	0

Figure 4: Compatible functions f_1 and f_2 for synchronous relation in Figure 3.

Definition 2 *For a given synchronous relation S, a function $f : B^m \rightarrow B^n$ is said to be* **compatible** *with S, denoted by $f \prec S$, iff for any d-vector of minterms $(\mathbf{x}^1, \ldots, \mathbf{x}^d) \in B^{md}$,*

$$S(\mathbf{x}^1, \ldots, \mathbf{x}^d, f(\mathbf{x}^1), \ldots, f(\mathbf{x}^d)) = 1$$

Example 2.2 *For the synchronous relation in Example 2.1, there are exactly two functions $f_1, f_2 : B^1 \rightarrow B^1$ which are compatible with $S(\mathbf{x}^1, \mathbf{x}^2, \mathbf{y}^1, \mathbf{y}^2)$. These two functions are shown in Figure 4.*

We have so far assumed that the depth of the input variables is the same as the depth of the output variables. However, we can show that if we have a synchronous relation where the input depth is larger than the output depth, we can construct an equivalent synchronous relation with equal input/output depth so that a compatible function with the new synchronous relation corresponds to a compatible function with the old relation. Consider the synchronous relation $S_1(\mathbf{x}^1, \ldots, \mathbf{x}^{d_1+d_2}, \mathbf{y}^1, \ldots, \mathbf{y}^{d_1}) \subseteq B^{m(d_1+d_2)} \times B^{nd_1}$. This synchronous relation relates the output variables over the last d_1 cycles to the input variables over the last $(d_1 + d_2)$ clock cycles. From S_1 we will construct a new synchronous relation S_2 between a new variable $\mathbf{z}$ and $\mathbf{y}$ such that the variable $\mathbf{z}$, at any time instant, is composed of the the values of $\mathbf{x}$ over that last $(d_2 + 1)$ clock cycles, i.e. $\mathbf{z}^i \equiv (\mathbf{x}^i, \ldots, \mathbf{x}^{i+d_2})$. Let $\mathbf{z}^i_j$ denote $\mathbf{x}^{i+j}$. The relation S_2 will have the same depth d_1 for $\mathbf{z}$ and $\mathbf{y}$. However, since each $\mathbf{z}^i$ represents $(d_2 + 1)$ time-shifted values of $\mathbf{x}$, each $\mathbf{x}^j$ may occur as many as $(d_2 + 1)$ times in S_2. We set the value of S_2 whenever all these repeated variables are equal; else, S_2 is 1 for all values of $(\mathbf{y}^1, \ldots, \mathbf{y}^{d_1})$ (essentially a don't care). In other words, S_2 corresponds to S_1 only when $(\mathbf{z}^i_j = \mathbf{z}^{i-k}_{j+k})$ for all valid values of i, j, k. If this condition is not true for any minterm $(\mathbf{z}^1, \ldots, \mathbf{z}^{d_1})$, we allow any value of $(\mathbf{y}^1, \ldots, \mathbf{y}^{d_1})$ for such a minterm by setting

$(\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3)$	$(\mathbf{y}^1, \mathbf{y}^2)$
(0,–,–)	{(1,0), (0,1)}
(1,–,1)	{(1,1)}
(1,0,0)	{(0,0)}
(1,1,0)	{(0,1), (1,1)}

Figure 5: Synchronous relation S_1 with unequal depths for $\mathbf{x}$ and $\mathbf{y}$. A – denotes a don't care value (can be either 0 or 1).

$S_2(\mathbf{z}^1, \ldots, \mathbf{z}^{d_1}, \mathbf{y}^1, \ldots, \mathbf{y}^{d_1})$ to be 1. Thus we define $S_2 \subseteq B^{m(d_2+1)d_1} \times B^{nd_1}$ as:

$$S_2(\mathbf{z}^1, \ldots, \mathbf{z}^{d_1}, \mathbf{y}^1, \ldots, \mathbf{y}^{d_1}) = \begin{cases} S_1(\mathbf{z}_0^1, \ldots, \mathbf{z}_0^{d_1}, & \mathbf{z}_1^{d_1}, \ldots, \mathbf{z}_{d_2}^{d_1}, \mathbf{y}^1, \ldots, \mathbf{y}^{d_1}) \\ & \text{if } \prod_{i=1}^{d_1} \prod_{j=0}^{d_2} \prod_{k=1}^{min(i-1, d_2-j)} (\mathbf{z}_j^i = \mathbf{z}_{j+k}^{i-k}), \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

In the above formulation, the limits for $\prod$ have been determined so that i and $(i-k)$ lie between 1 and d_1, and j and $(j+k)$ lie between 0 and d_2. Also note again that $\mathbf{z}_j^i$ denotes $\mathbf{x}^{i+j}$. It is easily seen that a function is compatible with S_2 iff an equivalent function is compatible with S_1. Unlike, the synchronous relations with equal depth, we will now have a solution for $\mathbf{y}$ which may depend upon the values of $\mathbf{x}$ at the previous d_2 clock cycles as well. So, in effect, we have a sequential implementation for the synchronous relation. However, the minimization procedure for finding a compatible function f remains unchanged.

Example 2.3 *Here, we show an example of the above construction to derive a synchronous relation of equal depths from a synchronous relation of unequal depths. Consider a synchronous relation* $S_1(\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \mathbf{y}^1, \mathbf{y}^2)$ *as shown in Figure 5. Here* $d_1 = 2, d_2 = 1$. *We will construct a synchronous relation* $S_2(\mathbf{z}^1, \mathbf{z}^2, \mathbf{y}^1, \mathbf{y}^2)$ *such that any compatible function for* S_2 *will represent a compatible function for* S_1. *Expression (1) transforms to:*

$$S_2(\mathbf{z}^1, \mathbf{z}^2, \mathbf{y}^1, \mathbf{y}^2) = \begin{cases} S_1(\mathbf{z}_0^1, \mathbf{z}_0^2, \mathbf{z}_1^2, \mathbf{y}^1, \mathbf{y}^2) & \textit{if } (\mathbf{z}_0^2 = \mathbf{z}_1^1), \\ 1 & \textit{otherwise} \end{cases} \quad (2)$$

The resulting synchronous relation S_2 *is shown in Figure 6.*

In our minimization procedure we need to check the compatibility of a function with a synchronous relation many times. A direct compatibility check, from Definition 2, requires the evaluation of the function f at each of the d minterms for every d-vector in the input space. To reduce the cost of the compatibility check, we form a new relation which will

$(\mathbf{z}^1, \mathbf{z}^2)$	$(\mathbf{y}^1, \mathbf{y}^2)$
(00,0–)	{(1,0), (0,1)}
(01,1–)	{(1,0), (0,1)}
(10,01)	{(1,1)}
(11,11)	{(1,1)}
(10,00)	{(0,0)}
(11,10)	{(0,1), (1,1)}
(–0,1–)	{(0,0), (0,1), (1,0), (1,1)}
(–1,0–)	{(0,0), (0,1), (1,0), (1,1)}

Figure 6: Synchronous relation S_2 with equal depths corresponding to synchronous relation S_1 in Figure 5. The last two rows correspond to the don't care condition in the construction of S_2.

enable us to do a compatibility check requiring the function evaluation only for a single minterm for every d-vector.

Definition 3 *A relation $R \subseteq B^{md} \times B^{nd}$ is called the* **nucleus** *of a synchronous relation S if $R(\mathbf{x}^1, \ldots, \mathbf{x}^d, \mathbf{y}^1, \ldots, \mathbf{y}^d) = 1$ if and only if for any $(\mathbf{u}^1, \ldots, \mathbf{u}^d) \in B^{md}$ there exists $(\mathbf{v}^1, \ldots, \mathbf{v}^d) \in B^{nd}$ which satisfies the following three conditions:*

1. $S(\mathbf{u}^1, \ldots, \mathbf{u}^d, \mathbf{v}^1, \ldots, \mathbf{v}^d) = 1$
2. $\forall j \in \{2, \ldots, d\} : (\mathbf{u}^j \equiv \mathbf{x}^1) \Rightarrow (\mathbf{v}^j \equiv \mathbf{y}^1)$
3. $\forall j \in \{1, \ldots, d\} : (\mathbf{u}^j \equiv \mathbf{x}^j) \Rightarrow (\mathbf{v}^j \equiv \mathbf{y}^j)$

The nucleus can be computed as

$$R(\mathbf{x}^1, \ldots, \mathbf{x}^d, \mathbf{y}^1, \ldots, \mathbf{y}^d) = C_{\mathbf{u}^1, \ldots, \mathbf{u}^d} S_{\mathbf{v}^1, \ldots, \mathbf{v}^d} \quad (S(\mathbf{u}^1, \ldots, \mathbf{u}^d, \mathbf{v}^1, \ldots, \mathbf{v}^d) \prod_{j=2}^{d} ((\mathbf{u}^j \not\equiv \mathbf{x}^1) + (\mathbf{v}^j \equiv \mathbf{y}^1)) \prod_{j=1}^{d} ((\mathbf{u}^j \not\equiv \mathbf{x}^j) + (\mathbf{v}^j \equiv \mathbf{y}^j)))$$

where $(\mathbf{u} \not\equiv \mathbf{x})$ is the characteristic function given by $\sum_{k=1}^{m} (u_k \oplus x_k)$ while $(\mathbf{u} \equiv \mathbf{x})$ denotes the complement of $(\mathbf{u} \not\equiv \mathbf{x})$.

Note that the nucleus is computed only once at the beginning of the minimization procedure. The compatibility of a function is checked as described in the following theorem.

Theorem 2.1 *A function $f : B^m \rightarrow B^n$ is a compatible implementation of a synchronous relation S if and only if for any $(\mathbf{x}^2, \ldots, \mathbf{x}^d) \in B^{m(d-1)}$, there exists $(\mathbf{y}^2, \ldots, \mathbf{y}^d) \in B^{n(d-1)}$ such that $R(\mathbf{x}, \mathbf{x}^2, \ldots, \mathbf{x}^d, f(\mathbf{x}), \mathbf{y}^2, \ldots, \mathbf{y}^d) = 1$ for every $\mathbf{x} \in B^n$, where R is the nucleus of S.*

Proof: [If]

Consider any arbitrary $(\mathbf{u}^1,\ldots,\mathbf{u}^d) \in B^{md}$. We have to prove that $S(\mathbf{u}^1,\ldots,\mathbf{u}^d,f(\mathbf{u}^1),\ldots,f(\mathbf{u}^d)) = 1$. By the assumption of the [If] part, there exists $(\mathbf{y}^2,\ldots,\mathbf{y}^d) \in B^{n(d-1)}$ such that $\forall \mathbf{x} \in B^n : R(\mathbf{x},\mathbf{u}^2,\ldots,\mathbf{u}^d,f(\mathbf{x}),f(\mathbf{y}^2),\ldots,f(\mathbf{y}^d)) = 1$. Let $\mathbf{x} = \mathbf{u}^i, i \in \{2,\ldots,d\}$. Using Definition 3, there exists $(\mathbf{v}^1,\ldots,\mathbf{v}^d) \in B^{nd}$ such that the following three conditions hold:

(i) $S(\mathbf{x},\mathbf{u}^2,\ldots,\mathbf{u}^d,\mathbf{v}^1,\ldots,\mathbf{v}^d) = 1$

(ii) $\forall j \in \{2,\ldots,d\} : (\mathbf{u}^j \equiv \mathbf{x}) \Rightarrow (\mathbf{v}^j \equiv f(\mathbf{x}))$

(iii) $\forall j \in \{2,\ldots,d\} : (\mathbf{v}^j \equiv \mathbf{y}^j)$

$\mathbf{x} = \mathbf{u}^i$ and (ii) $\Rightarrow \mathbf{v}^i = f(\mathbf{u}^i)$. Combining this with (iii) we get, $\mathbf{y}^i = f(\mathbf{u}^i)$. Thus $\forall \mathbf{x} \in B^n : R(\mathbf{x},\mathbf{u}^2,\ldots,\mathbf{u}^d,f(\mathbf{x}),f(\mathbf{u}^2),\ldots,f(\mathbf{u}^d)) = 1$. Using Definition 3, $S(\mathbf{u}^1,\ldots,\mathbf{u}^d,f(\mathbf{u}^1),\ldots,f(\mathbf{u}^d)) = 1$.

[Only if]

Consider any arbitrary $(\mathbf{x}^2,\ldots,\mathbf{x}^d) \in B^{m(d-1)}$. Let $\mathbf{y}^j = f(\mathbf{x}^j), j \in \{2,\ldots,d\}$. Given an arbitrary $\mathbf{x} \in B^n$, we have to prove that $R(\mathbf{x},\mathbf{x}^2,\ldots,\mathbf{x}^d,f(\mathbf{x}),\mathbf{y}^2,\ldots,\mathbf{y}^d) = 1$. Consider any arbitrary $(\mathbf{u}^1,\ldots,\mathbf{u}^d) \in B^{md}$. Let $\mathbf{v}^j = f(\mathbf{u}^j), j \in \{1,\ldots,d\}$. Now,

(i) Since $f \prec S$, $S(\mathbf{u}^1,\ldots,\mathbf{u}^d,\mathbf{v}^1,\ldots,\mathbf{v}^d) = 1$

(ii) $\forall j \in \{2,\ldots,d\} : (\mathbf{u}^j \equiv \mathbf{x}) \Rightarrow (\mathbf{v}^j = f(\mathbf{u}^j) = f(\mathbf{x}))$

(iii) $\forall j \in \{1,\ldots,d\} : (\mathbf{u}^j \equiv \mathbf{x}^j) \Rightarrow (\mathbf{v}^j = f(\mathbf{u}^j) = f(\mathbf{x}^j) = \mathbf{y}^j)$

Thus, from Definition 3, we have proved the required. ∎

Let $\mathcal{F}$ denote the a sum-of-products representation of the function f. Following are some definitions regarding $\mathcal{F}$ which we will use in our presentation.

Definition 4 *For a given synchronous relation S and a compatible representation $\mathcal{F}$, a cube $c \in \mathcal{F}$ is* **relatively prime** *in $\mathcal{F}$ if replacing c by any cube $\tilde{c} \supset c$ results in an incompatible representation. A compatible representation $\mathcal{F}$ is said to be* **relatively prime** *if every cube of $\mathcal{F}$ is relatively prime in $\mathcal{F}$.*

Definition 5 *For a given synchronous relation S and a compatible representation $\mathcal{F}$, a cube $c \in \mathcal{F}$ is* **irredundant** *in $\mathcal{F}$ if removing c from $\mathcal{F}$ destroys the compatibility of $\mathcal{F}$. A*

compatible representation $\mathcal{F}$ is said to be **irredundant** *if every cube of $\mathcal{F}$ is irredundant in $\mathcal{F}$.*

Definition 6 *For a given synchronous relation S and a compatible representation $\mathcal{F}$, a cube c^* is said to be a* **feasible cube** *for $c \in \mathcal{F}$ if $\mathcal{F} - \{c\} \cup \{c^*\}$ is compatible with S.*

We also borrow the following notation from [WB91].

Notation 2 *For a given representation $\mathcal{F}$, a function $f : B^m \rightarrow B^n$ is uniquely defined, where the algebraic expression of f is given by $\mathcal{F}$. Each cube $c \in \mathcal{F}$, represents a product term p, and is specified as a row vector with two parts, $c = [I(c)|O(c)]$. $I(c)_j$ is 0 if $\overline{x_j}$ appears in p, 1 if x_j appears in p, and 2 if $\overline{x_j}$ and x_j do not appear in p. $O(c)_j$ is 1 if p is present in the algebraic expression of $f^{(j)}$, else it is 0. $M(c)$ denotes the set of minterms that belong to the input space of cube c. The characteristic function $F(\mathbf{x}, \mathbf{y})$ for $\mathcal{F}$ is defined as $F = \{(\mathbf{x}, \mathbf{y}) \in B^m \times B^n \mid \mathbf{y} = f(\mathbf{x})\}$. In this report, we frequently need to talk about the representation for $\mathcal{F} - \{c\}$ where c is a cube in $\mathcal{F}$. We denote the characteristic function of this by $F_c(\mathbf{x}, \mathbf{y})$.*

2.2 Solution space for synchronous relations

The set of functions that may be compatible with a given synchronous relation spans a space which is larger than the space that existing combinational minimization methods can explore. There is a hierarchy of various multi-output minimization methods based on the solution space the desired function may lie in.

1. *Completely specified functions.* The function is exactly specified for each output, given any input minterm.

2. *Incompletely specified functions with don't care minterms.* The function is completely specified on a subset of input minterms and is allowed to take any value on the remaining minterms.

3. *Incompletely specified functions with different don't cares for different outputs.* Here, for any minterm, the don't cares may be specified for certain outputs only. The different outputs are still independent and can be minimized independently.

4. *Boolean relations.* Each input minterm is associated with a set of output minterms. A compatible function has to map each minterm to one of the values in the set. Unlike the previous cases, the various outputs are correlated. However, the input minterms are not correlated and any compatible function may independently choose the output minterms for two distinct minterms.

5. *Synchronous relations.* Unlike any of the above formulations, the choice of output value at an input minterm may constrain the choices for other minterms. A synchronous relation can be thought of as a boolean relation with constraints. Later in this section we will show that a synchronous relation can represent compatibility of a function with an arbitrary *set* of relations.

The following theorem shows that the solution space for synchronous relations encompasses the space for boolean relations, and hence, also the solution space covered by the first three techniques.

Theorem 2.2 *Given a boolean relation $R \in B^m \times B^n$, for any depth $d \geq 1$, there exists a synchronous relation $S \in B^{md} \times B^{nd}$ such that a function $f : B^m \rightarrow B^n$ is compatible with R if and only if f is compatible with S.*

Proof: Define synchronous relation S as $S(\mathbf{x}^1, \ldots, \mathbf{x}^d, \mathbf{y}^1, \ldots, \mathbf{y}^d) \equiv \prod_{i=i}^{d} R(\mathbf{x}^i, \mathbf{y}^i)$.

[If]

Suppose f is compatible with R. Consider any d-vector of minterms $(\mathbf{x}^1, \ldots, \mathbf{x}^d) \in B^{md}$. From the definition of compatibility for boolean relations, for all $i \in \{1, \ldots, d\}$, $R(\mathbf{x}^i, f(\mathbf{x}^i)) = 1$. Thus, $\prod_{i=1}^{d} R(\mathbf{x}^i, \mathbf{y}^i) = 1$, and hence $S(\mathbf{x}^1, \ldots, \mathbf{x}^d, f(\mathbf{x}^1), \ldots, f(\mathbf{x}^d)) = 1$. Hence, $f \prec S$.

[Only if]

Suppose $f \prec S$. Consider any minterm $\mathbf{x} \in B^m$. Since $f \prec S$, $S(\mathbf{x}, \ldots, \mathbf{x}, f(\mathbf{x}), \ldots, f(\mathbf{x})) = 1$. By the construction of S, $\prod_{i=i}^{d} R(\mathbf{x}, f(\mathbf{x})) = 1$. Therefore, $R(\mathbf{x}, f(\mathbf{x})) = 1$, proving the compatibility of f with R. ∎

Since synchronous relations appear to take input minterm correlations into account, one might speculate that the solution space for synchronous relations can be just an arbitrary set of functions. This leads to the following conjecture.

$\mathbf{x} \in B^2$	$f_1, f_2, f_3, g : B^2 \rightarrow B^1$			
	f_1	f_2	f_3	g
00	0	0	0	0
01	0	0	1	0
10	0	1	0	0
11	1	0	0	0

Figure 7: Example where a synchronous relation is not equivalent to a set of relations

Conjecture: *Given any arbitrary set of functions* $\mathcal{F} = \{f_1, \ldots, f_k\}$ *such that* $\forall i \in 1, \ldots, k, f_i : B^m \rightarrow B^n$, *for any depth* $d \geq 1$, *there exists a synchronous relation* $S \in B^{md} \times B^{nd}$ *such that* $f \prec S$ *if and only if* $f \in \mathcal{F}$. *Specifically,* S *in the previous statement is such that* $S(\mathbf{x}^1, \ldots, \mathbf{x}^d, \mathbf{y}^1, \ldots, \mathbf{y}^d) = 1$ *if and only if there exists* $i \in 1, \ldots, k$ *such that* $f_i(\mathbf{x}^j) = \mathbf{y}^j$ *for all* $j \in 1, \ldots, d$.

However, as the example 2.1 shows, the conjecture is false.

Example 2.4 *Consider synchronous relations of depth 2. Let* $\mathcal{F} = \{f_1, f_2, f_3 : B^2 \rightarrow B^1\}$ *be the set of functions defined in Figure 7. Let the synchronous relation* $S \subseteq B^{2\times 2} \times B^{1\times 2}$ *be such that* $S(\mathbf{x}^1, \mathbf{x}^2, \mathbf{y}^1, \mathbf{y}^2) = 1$ *iff* $\prod_{j=1}^{2}(f_i(\mathbf{x}^j) = \mathbf{y}^j)$ *for some* $i \in \{1, 2, 3\}$. *Let* $g : B^2 \rightarrow B^1$ *be the function as defined in Figure 7. Clearly,* $g \prec S$, *because for any two input minterms there exists an* f_i *which maps both to* 0. *Thus* $\forall \mathbf{x}^1, \mathbf{x}^2 \in B^2 : S(\mathbf{x}^1, \mathbf{x}^2, 0, 0) = 1$. *Thus it is not always possible to include a chosen set of functions without allowing some additional functions as well.*

Increasing the depth of the synchronous relation increases the sets of functions the synchronous relation can span. This is illustrated by the following two results.

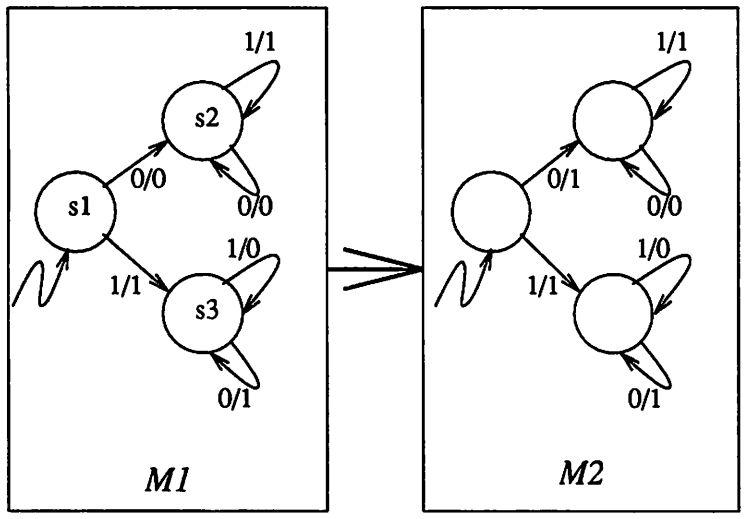
Theorem 2.3 *Suppose we are given any arbitrary set of boolean relations* $\mathcal{R} = \{R_1, \ldots, R_k\}$ *such that* $\forall i \in 1, \ldots, k, R_i \in B^m \times B^n$. *For any* $d \geq k$, *there exists a synchronous relation* $S \in B^{md} \times B^{nd}$ *such that* $f \prec S$ *if and only if* $f \prec R_i$ *for some* $R_i \in \mathcal{R}$.

Proof: We construct a synchronous relation S such that $S(\mathbf{x}^1, \ldots, \mathbf{x}^d, \mathbf{y}^1, \ldots, \mathbf{y}^d) = 1$ if and only if $\prod_{j=1}^{d} R_i(\mathbf{x}^j, \mathbf{y}^j) = 1$ for some $i \in \{1, \ldots, k\}$.

[If]

Obvious from the construction of S - follow the proof of Theorem 2.2.

[Only if]



(a)

PS	IN	OUT			
s1	0	0	0	1	1
s1	1	1	0	0	1
s2	0	0	0	1	1
s2	1	1	1	0	0
s3	0	1	0	0	1
s3	1	0	1	1	0

(b)

Figure 8: (a) Finite state machine $M1$ drives machine $M2$. (b) The specification for permitted output labels on machine $M1$, given that the behavior of the two-FSM system remains unchanged.

Suppose $f \prec S$. We want to show that $f \prec R_i$ for some $R_i \in \mathcal{R}$. Suppose not. Then, for all $i \in \{1, \ldots, k\}$, there exists $\mathbf{x}_i \in B^m$ such that $R_i(\mathbf{x}_i, f(\mathbf{x}_i)) = 0$. Consider the k-vector of these minterms, $(\mathbf{x}_1, \ldots, \mathbf{x}_k)$. If $d = k$, since $f \prec S$, $S(\mathbf{x}^1, \ldots, \mathbf{x}^d, f(\mathbf{x}^1), \ldots, f(\mathbf{x}^d)) = 1$. However, $\prod_{j=1}^{d} R_i(\mathbf{x}^j, \mathbf{y}^j) = 0$ for all $i \in \{1, \ldots, k\}$. This contradicts the definition of S. Hence, the proof follows for $d = k$. For $d > k$, the proof extends in an obvious way. ∎

Following the lines of the proof of Theorem 2.2, it is easy to show that synchronous relations of depth d_2 are strictly more expressive than synchronous relations of depth $d_1 < d_2$.

We now give an example of how synchronous relations arise naturally in the model of interacting FSMs. Consider the two FSM system in Figure 8(a), where $M1$ drives $M2$ (i.e. the output of $M1$ serves as the input for $M2$). Figure 8(b) shows the values the transition edge outputs edges can take without changing the behavior of the entire system ([WB93]). It can be verified that this table represents the *complete* characterization of the flexibility in choosing these output labels. Using Theorem 2.3, we can construct a synchronous relation of depth 4 which completely specifies the flexibility between the input space (PS, IN) and the output space OUT. However, since the output space is correlated with the input space (for example, $OUT = 0$ is related to $(PS, INP) = (s1, 0)$ if and only if $OUT = 0$ is related to $(PS, INP) = (s2, 0)$), this flexibility cannot be expressed by a single boolean relation over the space (PS, IN, OUT).

3 Minimization of compatible representation

3.1 Problem statement

The problem solved in this report is: *given a boolean synchronous relation* $S \subseteq B^{md} \times B^{nd}$, *find a minimal representation* $\mathcal{F}$ *which is relatively prime, irredundant, and compatible with* S.

We start with an initial representation $\mathcal{F}$ compatible with S. The cost function we use is the number of product terms in the representation. We then iteratively apply ESPRESSO-like procedures (REDUCE, EXPAND and IRREDUNDANT) to the sum of products until we cannot reduce the cost function further. The basic idea of ESPRESSO is to manipulate only one cube at a time and at all times the compatibility of the current representation with the synchronous relation is maintained; we adopt the same philosophy. It is thus very possible that our minimizer may get stuck in a local minimum. However, we guarantee that the representation at the end will be relatively prime and irredundant.

The REDUCE operation takes one cube c from the function representation $\mathcal{F}$ and replaces it by a minimum cube c^* so that $\mathcal{F} - \{c\} \cup \{c^*\}$ is compatible with the synchronous relation. The replacing cube is minimum in the sense that it is a smallest cube among the set of cubes that can be chosen to replace c. The replaced cube need not be contained in the original cube. EXPAND does the reverse operation - it expands a cube to a maximum cube such that it covers the most number of other cubes in $\mathcal{F}$. The expanded cube is guaranteed to be relatively prime. EXPAND (REDUCE) is iterated on all cubes of the representation, until no cube can be expanded (reduced) further. After EXPAND, the IRREDUNDANT operation is invoked. This just eliminates cubes which are irredundant, looking at one cube at a time.

3.2 Towards an initial representation

In this subsection we address the issue of finding an initial representation compatible with the synchronous relation. Unfortunately, we do not have an efficient method of solving this problem. However, in most cases where synchronous relation minimization is used to incrementally modify the design, as in Figure 2. Typically we *start* with a given initial compatible representation. Then we can use the minimization procedure of this report to exploit the flexibility of the synchronous relation.

One algorithm for obtaining an initial representation is presented below. Intuitively, we believe that algorithm is correct. However, we have not proved its correctness, because as it is, the algorithm is inefficient - it iterates B^m times over the inner loop. The *nucleus* operation in the following procedure is as defined in Definition 3.

```
procedure InitialRepresentation (S)
R ⇐ nucleus(S)
P(x, y) = S_{x^2,...,x^d,y^2,...,y^d} R(x, x^2, ..., x^d, y, y^2, ..., y^d)
F ⇐ φ
for all x̃ ∈ B^m do {
        do {
                S(x^1, ..., x^d, y^1, ..., y^d) = R(x^1, ..., x^d, y^1, ..., y^d) ∏_{j=1}^{d} P(x^j, y^j)
                R ⇐ nucleus(S)
                P(x, y) = S_{x^2,...,x^d,y^2,...,y^d} R(x, x^2, ..., x^d, y, y^2, ..., y^d)
        } until (R = S)
        Choose ỹ such that P(x̃, ỹ) = 1
        F ⇐ F + (x̃, ỹ)
        P(x, y) = P(x, y)((x ≡ x̃) ⇒ (y ≡ ỹ))
}
return F
```

Another worst-case exponential algorithm to obtain the initial representation can be obtained by using a brand-and-bound search. Although this may be exponential in the worst case, in practice it can yield fast solutions This has been shown to be the case in [KF92], where the objective is only to find an initial representation for a constrained boolean relation. It has been shown in [SSB93] that a constrained boolean relation is just another representation for a synchronous relation of a multiple boolean relation.

3.3 REDUCE

We define the maximally reduced cube just like for boolean relations ([WB91]).

Definition 7 *For a given representation $\mathcal{F}$ compatible with a synchronous relation S and a cube $c \in \mathcal{F}$, a cube $c^- \subseteq c$ is said to be the* **maximally reduced** *cube for c in $\mathcal{F}$ if $\mathcal{F} - \{c\} \cup \{c^-\}$ is compatible with S and there is no cube $\hat{c} \subset c^-$ such that $\mathcal{F} - \{c\} \cup \{\hat{c}\}$ is compatible with S.*

However, as we shall see in this section, our REDUCE procedure very naturally extends so that a cube need not be contained in the original cube. Rather, the smallest replacement cube can be found directly.

Definition 8 *Given $\mathcal{F}$, c and S as above, a cube c^* is said to be a* **maximally decreased** *cube for c in $\mathcal{F}$ if $\mathcal{F} - \{c\} \cup \{c^*\}$ is compatible with S and there is no other cube $\hat{c}$ smaller (has more literals) than c such that $\mathcal{F} - \{c\} \cup \{c^*\}$ is compatible with S.*

For incompletely specified functions (in ESPRESSO), there is a *unique* maximally reduced cube c^-. Also, for any cube $\hat{c}$ such that $c^- \subseteq \hat{c} \subseteq c$, replacing c by $\hat{c}$ preserves the compatibility of the representation (*continuity*). For boolean relations (in GYOCRO), these uniqueness and continuity hold for the input parts of the cube but are not true for the output parts. When these properties hold, c can be reduced one literal (input or output for ESPRESSO and input for GYOCRO) at a time and the maximally reduced cube is obtained. For GYOCRO, the cube can be reduced one literal at a time for the input part but must be further maximally reduced for the output part.

Since synchronous relations subsume boolean relations (as we showed in Theorem 2.2), the examples which show the non-uniqueness and non-continuity of REDUCE for output parts in boolean relations [WB91] hold for synchronous relations also. The following two examples illustrate the non-uniqueness and non-continuity for the input part.

Example 3.1 *Consider the two boolean relations $R_1, R_2 \subseteq B^2 \times B^1$ as shown in Figure 9. From Theorem 2.3 we can construct a synchronous relation $S \subseteq B^{2\times 2} \times B^{1\times 2}$ such that a function is compatible with S iff it is compatible with either R_1 or R_2. We start with representation $\mathcal{F} = \{x_1\}$ which is compatible with S. Now $\mathcal{F}$ can either be reduced to $\{x_0x_1\}$ or $\{\overline{x_0}x_1\}$, both of which are maximally reduced and are compatible with S. Thus the maximally reduced cube is not unique in the input part.*

Example 3.2 *Consider the synchronous relation S such that any $f : B^2 \rightarrow B^1$ is compatible with S iff f is compatible with either R_1 and R_2 as shown in Figure 10. Representation*

$\mathbf{x} \in B^2$	$R_1 \subseteq B^2 \times B^1$	$R_2 \subseteq B^2 \times B^1$
00	{0}	{0}
01	{0}	{1}
10	{0,1}	{0}
11	{1}	{0}

Figure 9: Example where a maximally reduced cube (input part) is not unique

$\mathbf{x} \in B^2$	$R_1 \subseteq B^2 \times B^1$	$R_2 \subseteq B^2 \times B^1$
00	{1}	{0}
01	{1}	{0}
10	{1}	{0}
11	{1}	{1}

Figure 10: Example where reduction is not continuous

$\mathcal{F} = \{1\}$ is compatible with S. Neither $\{x_0\}$ or $\{x_1\}$ is compatible with S. However, if we reduce further, $\{x_0 x_1\}$ is indeed compatible with S.

First we describe how to compute the maximally reduced cube c^- for $c \in \mathcal{F}$. We jump directly from c to c^- without going through any of the intermediate cubes.

Consider the cube c to be maximally reduced in an iteration of REDUCE. We wish to find the maximally reduced feasible cube for c. We will form a characteristic function $h(\mathbf{w}, \mathbf{u}, \mathbf{y})$ to represent the set of feasible cubes which are obtained by reducing c and maintain the compatibility of the function. Consider the sets $W_h = U_h = \{i \in \{1, \cdots, m\} \mid I(c)_i = 2\}$, and $Y_h = \{j \in \{1, \cdots, n\} \mid O(c)_j = 1\}$. Now, consider the boolean space $B^{|W_h|} \times B^{|U_h|} \times B^{|Y_h|}$ defined by the variables of W_h, U_h and Y_h. For each minterm $(\mathbf{w}, \mathbf{u}, \mathbf{y}) \in B^{|W_h|} \times B^{|U_h|} \times B^{|Y_h|}$ we define a reduced cube c^* as follows.

$$I(c^*)_i = \begin{cases} 2 & \text{if } i \in W_h, w_i = u_i = 1 \\ 1 & \text{if } i \in W_h, w_i = 0, u_i = 1 \\ 0 & \text{if } i \in W_h, w_i = 1, u_i = 0 \\ I(c)_i & \text{if } i \notin W_h \end{cases}$$

c^* is undefined if $w_i = u_i = 0$ for $i \in W_h$. $O(c^*)_j = 1$ if and only if $j \in Y_h$ and $y_j = 1$. Intuitively, w_i and u_i determine if the input part i is reduced or not, and if it is, the phase to which it is reduced. y_i determines if the output part i is reduced or not.

Example 3.3 *Consider the synchronous relation* $S \subseteq B^{2\times 2} \times B^{1\times 2}$ *described in Example 3.1. Consider the compatible representation* $\mathcal{F}$ *consisting of a single cube* $c = [12|1]$. *This cube can be reduced in the the input bit* x_2 *and also in the single output bit* y_1. *Thus, the sets* W_h, U_h *and* Y_h *are* $\{2\}$, $\{2\}$ *and* $\{1\}$, *respectively. Using the above definition for the reduced cube* c^*, *the minterm* $(w_2, u_2, y_1) = (0, 1, 0)$ *corresponds to* $c^* = [11|0]$.

Now let $h : B^{|W_h|} \times B^{|U_h|} \times B^{|Y_h|} \rightarrow B$ be a function such that $h(\mathbf{w}, \mathbf{u}, \mathbf{y}) = 1$ if and only if the reduced cube c^* is feasible. Thus, $h(\mathbf{w}, \mathbf{u}, \mathbf{y}) = 1$ if and only if $f \prec S$ where for each minterm $\mathbf{x} \in B^m$, $f(\mathbf{x}) = \mathbf{z}$ and $\mathbf{z} \in B^n$ is defined as:

$$\forall j \in \{1, \ldots, m\} : z_j = \begin{cases} r_j + y_j & \text{if } \mathbf{x} \in M(c^*), j \in Y_h, \\ r_j & \text{otherwise} \end{cases} \tag{3}$$

where the minterm $\mathbf{r} \in B^n$ such that $F_c(\mathbf{x}, \mathbf{r}) = 1$.

Example 3.4 *Consider, once again, the synchronous relation constructed in Example 3.1. We will describe the function* $h(\mathbf{w}, \mathbf{u}, \mathbf{y})$ *for the cube* $c = [12|1]$ *with respect to the single-cube compatible representation* $\mathcal{F} = \{x_1\}$. *From Figure 9, it is clear that the only feasible reductions to the cube are* $c^* = c$, $c^* = [10|1]$ *and* $c^* = [11|1]$. *Thus* $h(w_2, u_2, y_1) = 1$ *if and only if* $(w_2, u_2, y_1) \in \{(1,1,1), (1,0,1), (0,1,1)\}$.

Theorem 2.1 gives the conditions under which $f \prec S$ is true. Using the theorem, $h(\mathbf{w}, \mathbf{u}, \mathbf{y}) = 1$ if and only if for any $(\mathbf{x}^2, \ldots, \mathbf{x}^d) \in B^{m(d-1)}$, there exists $(\mathbf{y}^2, \ldots, \mathbf{y}^d) \in B^{n(d-1)}$ such that for all $\mathbf{x} \in B^m$, there exist $\mathbf{z}, \mathbf{r} \in B^n$ which satisfy the following three conditions:

1. $R(\mathbf{x}, \mathbf{x}^2, \ldots, \mathbf{x}^d, \mathbf{z}, \mathbf{y}^2, \ldots, \mathbf{y}^d) = 1$
2. $F_c(\mathbf{x}, \mathbf{r}) = 1$
3. $\mathbf{z}$ is defined as in (3).

Now, using the definition of $I(c^*)$, the function h can be computed using BDDs as

$$h(\mathbf{w}, \mathbf{u}, \mathbf{y}) = \mathcal{C}_{\mathbf{x}^2,\ldots,\mathbf{x}^d} \mathcal{S}_{\mathbf{y}^2,\ldots,\mathbf{y}^d} \mathcal{C}_{\mathbf{x}} \mathcal{S}_{\mathbf{z},\mathbf{r}} (R(\mathbf{x}, \mathbf{x}^2, \ldots, \mathbf{x}^d, \mathbf{z}, \mathbf{y}^2, \ldots, \mathbf{y}^d) F_c(\mathbf{x}, \mathbf{r}) H(\mathbf{w}, \mathbf{u}, \mathbf{y}, \mathbf{x}, \mathbf{z}, \mathbf{r}))$$

where H specifies condition (3) above and is computed as

$$\begin{aligned} H(\mathbf{w}, \mathbf{u}, \mathbf{y}, \mathbf{x}, \mathbf{z}, \mathbf{r}) \;=\; & \textstyle\prod_{i \in W_h} (w_i \overline{\oplus} x_i + u_i x_i) \prod_{i \notin W_h} (x_i \overline{\oplus} I(c)_i) \prod_{j \in Y_h} (z_j \overline{\oplus} (r_j + y_j)) \prod_{j \notin Y_h} (z_j \overline{\oplus} r_j) \\ & + \left(\textstyle\sum_{i \in W_h} ((x_i \overline{\oplus} w_i)(x_i \oplus u_i)) + \sum_{i \notin W_h} (x_i \oplus I(c)_i)\right) \prod_{\forall j} (z_j \overline{\oplus} r_j) \end{aligned}$$

Once we calculate $h(\mathbf{w},\mathbf{u},\mathbf{y})$, the maximally reduced cube corresponds to the minterm $(\mathbf{w},\mathbf{u},\mathbf{y})$ of h which has the minimum number of 1's. This is because we have defined c^* for a $(\mathbf{w},\mathbf{u},\mathbf{y})$ is such a way that a cube with more 1's is larger than a cube with fewer 1's. A minterm with a minimum number of 1's is given by a shortest path connecting the 1-leaf to the root of the BDD for $h(\mathbf{w},\mathbf{u},\mathbf{y})$, where the length of a BDD edge is 1 if the edge is the *then*-edge and 0 is the edge is the *else*-edge. This follows from a proof given in [LS90].

Maximally decreased cube

The formulation described in this section to obtain a maximally reduced cube can be easily extended to obtain a maximally decreased cube. Here we relax the restriction that the replacing cube is contained in the replaced cube. To get the maximally decreased cube, we only change the definitions for W_h, U_h and Y_h, and then use exactly the same method as above. The new definitions are: $W_h = U_h = \{1,\ldots,m\}$ and $Y_h = \{1,\ldots,n\}$. Effectively we start from a tautology cube (instead of c) and then maximally reduce it. This modification can produce a better solution to our problem but at the cost of extra processing.

3.4 EXPAND procedure

The aim of EXPAND is to replace a cube by a maximally expanded cube defined as follows.

Definition 9 *For a given representation $\mathcal{F}$ compatible with a synchronous relation S and a cube $c \in \mathcal{F}$, a cube $c^+ \supseteq c$ is said to be the* **maximally expanded** *cube for c in $\mathcal{F}$ if $\mathcal{F} - \{c\} \cup \{c^+\}$ is compatible with S and the following two conditions hold:*

1. *if $\sigma(c^+) \subseteq \mathcal{F} - \{c\}$ denotes the maximal set of cubes such that $\mathcal{F} - \{c\} - \sigma(c^+) \cup \{c^+\}$ is compatible, then there exists no cube $\hat{c} \supseteq c$ such that $\mathcal{F} - \{c\} \cup \{\hat{c}\}$ is compatible and $|\sigma(\hat{c})| > |\sigma(c^+)|$,*
2. *there exists no cube $\hat{c} \supset c^+$ such that $\mathcal{F} - \{c\} \cup \{\hat{c}\}$ is compatible.*

Definition 10 *Given $\mathcal{F}$, c and S as above, a cube c^* is said to be a* **maximally increased** *cube for c in $\mathcal{F}$ if $\mathcal{F} - \{c\} \cup \{c^*\}$ is compatible with S and the following two conditions hold:*

1. *if $\sigma(c^*) \subseteq \mathcal{F} - \{c\}$ denotes the maximal set of cubes such that $\mathcal{F} - \{c\} - \sigma(c^*) \cup \{c^*\}$ is compatible, then there exists no cube $\hat{c} \supseteq c$ such that $\mathcal{F} - \{c\} \cup \{\hat{c}\}$ is compatible and $|\sigma(\hat{c})| > |\sigma(c^*)|$,*

2. *there exists no cube $\hat{c}$ larger (has fewer literals) than c^* such that $\mathcal{F} - \{c\} \cup \{\hat{c}\}$ is compatible and $|\sigma(\hat{c})| \geq |\sigma(c^*)|$.*

Replacing a cube by a maximally expanded or maximally increased cube, therefore, guarantees that a maximal number of cubes are covered by the new cube and the new cube is relatively prime with respect to the representation.

The EXPAND procedure follows on similar lines as ESPRESSO and GYOCRO. We have a covering matrix C which represents the rest of the cubes in the representation. This matrix is used to guide the expansion in a direction which facilitates the removal of as many cubes as possible. We maintain two sets, $\mathcal{R}$ and $\mathcal{L}$ which respectively denote the literals that have been raised and the literals that have been determined not to be raised.

Now we describe the operations used in EXPAND in some detail. For a more detailed description the reader is referred to [BHMS84, WB91]. The ESSENTIAL operation finds literals which cannot be raised during the current expansion and adds them to $\mathcal{L}$. Cubes in C which are impossible to be covered now are removed. The key operation in EXPAND is MFC. This determines the directions for the expansion of the cube. As in GYOCRO, some approximations are introduced in MFC to speed up the computation for the procedure. First, the set $\sigma(c)$, introduced in Definition 9, is computed greedily and not exactly. For each cube p in C, we form a cube $\hat{c}$ by taking the partwise union of p and $\mathcal{R}$. Then we calculate a maximally expanded cube c^* for $\hat{c}$ using the procedure described in the next subsection. p is selected such that the maximum possible cubes in C are covered by c^*, and $\mathcal{R}$ is updated to reflect the replacement of the current cube by c^*. This procedure is iterated until no such p can be selected for which c^* will cover any more cubes in C. At this point, we greedily expand the cube to a maximal feasible cube so that it is relatively prime.

Forming an expanded feasible cube

We describe how to form a maximally expanded cube $c^* \supseteq \hat{c}$ starting from $\hat{c}$. We use a technique similar to the one used for REDUCE. Unlike REDUCE, we will need only two variables, $\mathbf{w}$ and $\mathbf{y}$, to represent the characteristic function for all feasible cubes obtained by expanding $\hat{c}$.

Using $\mathcal{L}$, the lowering set defined earlier, we define the following sets:

$$W_k \;=\; \{i \in \{1, \cdots, m\} \mid I(\hat{c})_i \neq 2, i \notin \mathcal{L}\}$$

$$
\begin{aligned}
W_l &= \{i \in \{1,\cdots,m\} \mid i \in \mathcal{L}\} \\
Y_k &= \{j \in \{1,\cdots,n\} \mid O(\hat{c})_j = 0, (j+m) \notin \mathcal{L}\} \\
Y_l &= \{j \in \{1,\cdots,n\} \mid (j+m) \in \mathcal{L}\}
\end{aligned}
$$

Consider the boolean space $B^{|W_k|} \times B^{|Y_k|}$ defined by the variables of W_k and Y_k. For each minterm $(\mathbf{w},\mathbf{y}) \in B^{|W_k|} \times B^{|Y_k|}$, we define an expanded cube c^* as follows.

$$
I(c^*)_i = \begin{cases} 2 & \text{if } (i \in W_k, w_i = 1) \text{ or } (i \notin (W_k \cup W_l)) \\ I(c)_i & \text{otherwise} \end{cases}
$$

$O(c^*)_j = 1$ if and only if either $(j \in Y_k, y_j = 1)$ or $(j \notin Y_k \cup Y_l)$. Let $k : B^{|W_k|} \times B^{|Y_k|} \rightarrow B$ be a function such that $k(\mathbf{w},\mathbf{y}) = 1$ if and only if $\mathcal{F} - \{c\} \cup \{c^*\}$ is compatible with the synchronous relation. This means that $k(\mathbf{w},\mathbf{y}) = 1$ if and only if $f \prec S$, where $f : B^m \rightarrow B^n$ is defined so that for each minterm $\mathbf{x} \in B^m$, $f(\mathbf{x}) = \mathbf{z}$ and $\mathbf{z} \in B^n$ is specified as:

$$
\forall j \in \{1,\ldots,m\} : z_j = \begin{cases} r_j + y_j & \text{if } \mathbf{x} \in M(c^*), j \in Y_k, \\ 1 & \text{if } \mathbf{x} \in M(c^*), j \notin (Y_k \cup Y_l), \\ r_j & \text{otherwise} \end{cases} \tag{4}
$$

where the minterm $\mathbf{r} \in B^n$ is such that $F_c(\mathbf{x},\mathbf{r}) = 1$.

Using Lemma 2.1 in a way similar to the formulation of $h(\mathbf{w},\mathbf{u},\mathbf{y})$ in REDUCE, we can formulate the computation of k as:

$$
k(\mathbf{w},\mathbf{y}) = \mathcal{C}_{\mathbf{x}^2,\ldots,\mathbf{x}^d}\mathcal{S}_{\mathbf{y}^2,\ldots,\mathbf{y}^d}\mathcal{C}_{\mathbf{x}}\mathcal{S}_{\mathbf{z},\mathbf{r}}(R(\mathbf{x},\mathbf{x}^2,\ldots,\mathbf{x}^d,\mathbf{z},\mathbf{y}^2,\ldots,\mathbf{y}^d)F_c(\mathbf{x},\mathbf{r})K(\mathbf{w},\mathbf{y},\mathbf{x},\mathbf{z},\mathbf{r})) \tag{5}
$$

where K computes the relation between $(\mathbf{w}, \mathbf{y}, \mathbf{x}, \mathbf{z}, \mathbf{r})$ as defined in (4). K is computed as:

$$
\begin{aligned}
K(\mathbf{w},\mathbf{y},\mathbf{x},\mathbf{z},\mathbf{r}) = {} & \textstyle\prod_{i\in W_k}(w_i + (x_i \overline{\oplus} I(\hat{c})_i)) \prod_{i\in W_l}(x_i \overline{\oplus} I(\hat{c})_i) \prod_{j\in Y_k}(z_j \overline{\oplus}(r_j + y_j)) \prod_{j\in Y_l}(z_j \overline{\oplus} r_j) \prod_{j\notin(Y_k\cup Y_l)}(z_j) \\
& + (\textstyle\sum_{i\in W_k}(\overline{w}_i(x_i \oplus I(\hat{c})_i)) + \sum_{i\in W_l}(x_i \oplus I(\hat{c})_i)) \prod_{\forall j}(z_j \overline{\oplus} r_j)
\end{aligned}
$$

If $k(\mathbf{w},\mathbf{y}) \equiv 0$, then there exists no feasible cube for $\hat{c}$. Otherwise, like in REDUCE, the smallest feasible cube for $\hat{c}$ is given by the minterm $(\mathbf{w},\mathbf{y})$ of k which has the fewest 1's. Such a minterm is found by performing a shortest path algorithm for the BDD of $k(\mathbf{w},\mathbf{y})$ just as in REDUCE.

At the end of the expansion for each cube c, we would like to maximally expand the current cube to make it relatively prime. So we have to find a maximal feasible cube now. In order to do this, we do the same computation as above, except that we compute the *longest* path in the BDD instead.

A performance optimization

In each EXPAND loop (while expanding a particular cube for a particular representation), we try to expand the cube many times - each time we try to cover another cube in the covering matrix, we raise certain columns of $\mathcal{R}$ and try to find a feasible cube, and each time we try to add more columns to $\mathcal{L}$ while doing ESSENTIAL, we raise one column of $\mathcal{R}$ and see if the column is essential or not.

It is expensive to compute the function $k(\mathbf{w},\mathbf{y})$ each time we try to expand the cube represented by $\mathcal{R}$. So we compute this function only once for each cube expansion relative to a particular representation $\mathcal{F}$. At the beginning of the loop we compute $k(\mathbf{w},\mathbf{y})$ and then iteratively modify it. As we enter the loop, $\mathcal{R}$ is set to the the cube c, and $\mathcal{L}$ is set to the null set. We, thus compute the initial $k(\mathbf{w},\mathbf{y})$ using the value of $\hat{c} = c$. Later, whenever we move a literal to $\mathcal{R}$ or $\mathcal{L}$, we use the following rules to modify k:

- If input literal i is moved to $\mathcal{R}$, cofactor $k(\mathbf{w},\mathbf{y})$ with w_i. If output literal j is moved to $\mathcal{R}$, cofactor $k(\mathbf{w},\mathbf{y})$ with y_j.
- If input literal i is moved to $\mathcal{L}$, cofactor $k(\mathbf{w},\mathbf{y})$ with $\overline{w_i}$. If output literal j is moved to $\mathcal{L}$, cofactor $k(\mathbf{w},\mathbf{y})$ with $\overline{y_j}$.

One can easily verify the validity of the above rules from the expression for the BDD computation of $k(\mathbf{w},\mathbf{y})$. Using this optimization gave us a speedup of 2 on some examples over the naive implementation obtained by a direct modification of the GYOCRO code.

Maximally increased cube

As in REDUCE, we would like to remove the restriction about the relation between the cube to be modified and the modified cube. Here, we remove the restriction that the expanded cube contain original cube. We will, therefore, look for a maximally increased cube instead of a maximally expanded cube. This is accomplished by using $\mathcal{R} = \phi$ instead of c at the start of the cube expansion for a particular cube and a particular representation.

3.5 IRREDUNDANT procedure

The IRREDUNDANT procedure looks at one cube $c \in \mathcal{F}$ at a time. The cube is removed from the representation if the function represented by $\mathcal{F} - \{c\}$ is compatible with the synchronous

relation.

4 Results

We have implemented two versions of the minimizer. The first produces a minimally reduced cube in REDUCE and a maximally expanded cube in EXPAND. In the second version, we do not restrict EXPAND (REDUCE) in order that the replacing cube contains (is contained in) the original cube. The two programs that implement these versions are called CHAI-REI and CHAI-IDI, respectively.

Since no synchronous relation benchmarks exist, we constructed a few synchronous relations by composing boolean relations as described in Lemma 2.2. Our minimizer was able to jump from one relation to another in search for a smaller sum of products representation.

We performed an experiment to compare the computational efficiency of CHAI-x with a boolean relation minimizer GYOCRO([WB91]). We composed each of our synchronous relations from a single boolean relation using the method described in the Lemma 2.2. We chose the depth of the synchronous relation to be 2. So given the boolean relation $R(\mathbf{x},\mathbf{y})$, we construct a synchronous relation S such that $S(\mathbf{x}^1,\mathbf{x}^2,\mathbf{y}^1,\mathbf{y}^2)$ is 1 if and only if $R(\mathbf{x}^1,\mathbf{y}^1) = R(\mathbf{x}^2,\mathbf{y}^2) = 1$. It must be noted that in composing the synchronous relation this way, we double the number of input and output variables and square the number of minterms in the input relation specification. For the initial representation, we used the initial representation generated by GYOCRO. Note that we could have used the techniques we discussed in Section 3.2 to obtain an initial representation. We selected the example circuits where GYOCRO did not give the exact minimum representation to see if CHAI does better.

The purpose of the experiment is to display the comptutational efficiency of CHAI. It is not meant to be faster or better than GYOCRO for solving ordinary boolean relations. Since the depth-2 sychronous relations for the experiments lie in a much larger space than the boolean relations they are derived from, and CHAI uses the same basic approach as GYOCRO, we expect CHAI to take much larger times and give almost identical solutions as GYOCRO. However, through our experiments, we wish to show that computational times for CHAI are not many orders of magnitude larger than those for GYOCRO. On the other hand, GYOCRO is only a boolean relation minimizer, cannot minimize arbitrary synchronous relations whereas

Name	I/O	Input PLA	GYOCRO #	GYOCRO Time	CHAI-REI #	CHAI-REI Time	CHAI-IDI #	CHAI-IDI Time
int4	4/3	28	9	0.66	7	3.56	7	8.96
int8	4/3	25	9	1.53	8	4.92	8	10.96
c17c	5/3	59	16	2.33	16	19.26	16	39.37
c17d	5/3	54	16	2.21	15	33.54	15	76.75
c17e	5/3	41	5	0.91	5	4.55	5	9.44
c17g	5/3	82	7	1.32	7	4.54	7	7.82
c17i	5/3	43	15	1.79	15	13.00	15	29.33
she2	5/5	56	10	7.55	11	44.85	10	82.81
she4	5/6	97	20	16.28	20	122.03	20	238.03
b9	16/5	186	270	300.0	?	?	?	?

Table 1: Implementation results for CHAI

CHAI is a synchronous relation minimizer which can minimize arbitrary boolean relations as well. Table 1 shows our experiments. The table shows the number of product terms in the minimized representation and CPU time (in seconds) required on a DECstation 5900 for each of GYOCRO, CHAI-REI and CHAI-IDI. CHAI-REI is about $7-8$ times slower than GYOCRO; CHAI-IDI is about 2 times slower than CHAI-REI. Our minimizer could not even read the input relation for the circuit b9. This example had 68940 product terms in the PLA for the synchronous relation and we could not even construct the input synchronous relation, possibly because of a poor BDD variable ordering for the 42 variables (16 input and 5 output for each depth). In Table 1, The number of products terms are slightly different for GYOCRO and CHAI-REI for a few examples because GYOCRO expands the input and output parts separately and thus may not actually get the maximum reduction. Our implementations minimize both the input and the output parts together, albeit at a higher processing cost.

There are two reasons why CHAI is slower than GYOCRO. First, the check for compatibility of a function (Theorem 2.1) is much more expensive - in GYOCRO, the compatibility check is just a containment check. Second, CHAI cannot manipulate the input space one literal a time as GYOCRO is able to do. There is one place where CHAI is able to perform better than GYOCRO - in EXPAND, the function $k(\mathbf{w},\mathbf{y})$ has to be computed only once for each cube (relative to a given $\mathcal{F}$). Thereafter, it can be iteratively modified by taking cofactors. In GYOCRO, on the other hand, this cannot be done - only a smaller part of the equivalent function can be iteratively modified.

5 Conclusions

In this report we have shown the expressive power of synchronous relations and have shown that they are more expressive than ordinary boolean relations. We have then presented a heuristic method to find a minimal implementation for a synchronous relation iterating from a given initial representation. We have also shown that the computing times for our minimizer are feasible for small examples.

Forming an initial compatible representation efficiently still needs to be solved. However, an initial representation need not be recomputed if one already starts with a given design which is to be optimized.

A future line of research is to explore the ways in which synchronous relations can be formed for a given system. It will also be useful to find out how sequential circuits should be partitioned in order to get the maximum advantage from our minimizer.

Acknowledgements

We wish to thank Hervé Touati for discussions on the formulation of the *nucleus* synchronous relation, Yuji Kukimoto for suggestions on forming initial representation, and Huey Wang for the example of synchronous relation in interacting FSMs.

References

[BHMS84] R.K. Brayton, G.D. Hachtel, C.T. McMullen, and A.L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, 1984.

[Bry86] R. Bryant. Graph-based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35:677–691, August 1986.

[BS89] R.K. Brayton and F. Somenzi. Minimization of Boolean Relations. In *Proceedings of the International Symposium on Circuits and Systems*, May 1989.

[DDM92a] M. Damiani and G. De Micheli. Synthesis and Optimization of Synchronous Logic Circuits from Recurrence Equations. In *Proceedings of the European Conference on Design Automation*, pages 226–231, March 1992.

[DDM92b] M. Damiani and G. De Micheli. Recurrence Equations and the Optimization of Synchronous Logic Circuits. In *Proceedings of the Design Automation Conference*, pages 556–561, June 1992.

[KF92] Y. Kukimoto and M. Fujita. Rectification Method for Lookup-Table Type FPGA's. In *Proceedings of the International Conference on Computer-Aided Design*, pages 54–61, November 1992.

[LS90] B. Lin and F. Somenzi. Minimization of Symbolic Relations. In *Proceedings of the International Conference on Computer-Aided Design*, pages 88–91, November 1990.

[RHS91] J.-K. Rho, G. Hachtel, and F. Somenzi. Don't Care Sequences and the Optimization of Interacting Finite State Machines. In *Proceedings of the International Conference on Computer-Aided Design*, pages 418–421, November 1991.

[SSB93] E.M. Sentovich, V. Singhal, and R.K. Brayton. Multiple Boolean Relations. In *Proceedings of the International Workshop on Logic Synthesis*, May 1993.

[WB91] Y. Watanabe and R.K. Brayton. Heuristic Minimization of Multiple-Valued Relations. In *Proceedings of the International Conference on Computer-Aided Design*, pages 126–129, November 1991.

[WB93] H-Y. Wang, R.K. Brayton. Permissible Observability Relations in Interacting Finite State Machines. In *Proceedings of the International Workshop on Logic Synthesis*, May 1993.