Hindawi
International Journal of Reconfigurable Computing
Volume 2019, Article ID 3727254, 17 pages
https://doi.org/10.1155/2019/3727254

Research Article

FPGA Implementation of an Improved Reconfigurable FSMIM Architecture Using Logarithmic Barrier Function Based Gradient Descent Approach

Nitish Das and Aruna Priya P

Department of ECE, SRM Institute of Science & Technology, Kattankulathur-603203, Chennai, India

Correspondence should be addressed to Aruna Priya P; arunapriya.p@ktr.srmuniv.ac.in

Received 31 October 2018; Revised 2 January 2019; Accepted 10 January 2019; Published 1 April 2019

Academic Editor: John Kalomiros

Recently, the Reconfigurable FSM has drawn the attention of the researchers for multistage signal processing applications. The optimal synthesis of Reconfigurable finite state machine with input multiplexing (Reconfigurable FSMIM) architecture is done by the iterative greedy heuristic based Hungarian algorithm (IGHA). The major problem concerning IGHA is the disintegration of a state encoding technique. This paper proposes the integration of IGHA with the state assignment using logarithmic barrier function based gradient descent approach to reduce the hardware consumption of Reconfigurable FSMIM. Experiments have been performed using MCNC FSM benchmarks which illustrate a significant area and speed improvement over other architectures during field programmable gate array (FPGA) implementation.

1. Introduction

Digital signal processing (DSP) [1–3], pattern matching [4], and circuit testing [5] are the primary applications for most of the digital systems. These applications require a hardware-oriented as well as high-speed control unit. A finite state machine (FSM) is an integral part of any complex digital system. Its inputs are multiplexed to make it hardware oriented, which is known as the finite state machine with input multiplexing (FSMIM). It serves as a control unit, and its operating speed determines the processing speed of the system. The applications as mentioned earlier can be observed as cascaded stages (i.e., multistage) of operations [2], where each stage requires a specific FSM. Hence, a Reconfigurable FSM is investigated in the literature for optimal performance in such applications [6, 7]. A Reconfigurable FSM is defined as a single FSM, which acts as one of the FSMs from the set (i.e., set of FSMs for a specific application) by applying particular mode bits. Its implementation is performed on field programmable gate array (FPGA) platforms [6].

The Reconfigurable FSMIM architecture is created by joining (A) Conventional FSMIM architecture [8] and (B) multiplexer bank (which defines the mode based reconfiguration). The optimal synthesis of both the constituting elements is done by Iterative greedy heuristic based Hungarian algorithm (IGHA) [6]. An efficient state encoding technique for an FSM serves as a vital tool to optimize the hardware utilization while implementing on an FPGA platform [9, 10]. In the case of Reconfigurable FSMIM, the state encoding of the constituent FSMs altogether affects the look-up table (LUT) requirement of the Reconfigurable FSMIM [6].

The major problem concerning IGHA is the disintegration of a state encoding technique. It uses binary state encoding as a default state assignment technique for operation. The state assignment method for the Reconfigurable FSMIM architecture leads to an optimization problem [6]. To the best of the authors' knowledge, all the state assignment techniques proposed in the literature provide state codes only for a single FSM. Therefore, the objective of this work is the integration of IGHA with an optimal state encoding technique to reduce the hardware consumption of Reconfigurable FSMIM on an FPGA platform.

In the literature, another direction in the implementation of an FSM is RAM-based architectures. The following three

types of RAM-based FSM architectures are studied [11]: (a) basic RAM-based FSM architecture, (b) RAM-based FSM architecture with transition-controlled multiplexers, and (c) RAM-based FSM architecture with state-controlled multiplexers. In the basic RAM-based FSM architecture, bits are stored in the form of words. For each transition (i.e., present state combined with the external inputs), the outputs and the state assignment bits for next state are stored in the RAM-word memory [12, 13]. The RAM size required for basic RAM-based FSM implementation is enormous. Hence, to reduce the RAM depth, RAM-based FSM architecture with transition-controlled multiplexers is used. It consists of an input selector bank, which provides active inputs from the external inputs for selecting a particular state [11]. RAM-based FSM architecture with state-controlled multiplexers is used to reduce the RAM size further. It consists of two separate RAM blocks, out of which the smaller RAM block is assigned to operate the input selector bank [11]. Thus, designing such architecture is very complicated.

In this paper, the Improved Reconfigurable FSMIM architecture is proposed, which surmounts the issue of high LUT consumption during FPGA implementation. The proposed architecture is formed using the improved iterative greedy heuristic based Hungarian algorithm (Improved-IGHA). The Improved-IGHA is the integration of IGHA with the state assignment using logarithmic barrier function based gradient descent approach.

To validate the proposed approach, experiments have been performed using MCNC FSM benchmarks [14]. Experimental results for the proposed architecture illustrate a significant area reduction by an average of 20.38% and speed improvement by an average of 32.73% over VRMUX [11] during FPGA implementation. It also demonstrates an adequate area reduction by an average of 16.05% and speed improvement by an average of 1.77% over Reconfigurable FSMIM-S architecture [6] during FPGA implementation. When these results are compared with CRMUX [11], a speed improvement by an average of 11.06% is obtained. The proposed architecture requires an average of 58.38% more LUTs as compared with CRMUX [11] during FPGA implementation. It is the only trade-off for the proposed design.

The remainder of this article is formed as follows. The research problem formulation is made in Section 2. Section 3 consists of state assignment using logarithmic barrier function based gradient descent approach and an illustrative example. Experimental setup and comparative analysis of this work with the literature are devised in Section 4. In the end, concluding remarks are drawn in Section 5.

2. Problem Formulation

Recently, the Reconfigurable FSM has drawn the attention of the researchers for multistage signal processing applications. A novel framework for the creation of Reconfigurable FSMIM is given in [6].

A Mealy FSM is represented in a vector form, such as $(S, X, Y, \delta, \pi, S0)$ where

$S = (S0, \ldots, S(M)) \longleftarrow$ set of states;

$X = (x_1, \ldots, x_L) \longleftarrow$ set of input variables;

$Y = (y_1, \ldots, y_N) \longleftarrow$ set of output variables;

$\delta \Longrightarrow S * X \longrightarrow S \longleftarrow$ transition function;

$\pi \Longrightarrow S * X \longrightarrow Y \longleftarrow$ output function;

$S0 \longleftarrow$ initial state.

Moreover, the following variables are defined to illustrate the complete functionality of an FSM:

$S(m) \longleftarrow$ any instantaneous state $S(m) \in S$ where $m \in (0, 1, \ldots, M)$;

$K(S(m)) \longleftarrow$ binary state code for the, state $S(m) \in S$;

$H = (t_1, \ldots, t_M) \longleftarrow$ set of number of transitions per state corresponding to S;

$h \longleftarrow$ number of transitions per state where $h \in (1, 2, \ldots, H)$;

$R \longleftarrow$ the minimum length of a binary-state code, $R = \lceil \log_2 M \rceil$.

The Reconfigurable FSMIM is defined as a single FSM, which acts as any one of the FSM from the set (i.e., set of FSMs for a specific application) by applying particular mode bits. A set of FSM for a specific application is chosen, where *base_ckt* $\longleftarrow$ the largest FSM (i.e., the FSM with the highest total number of transitions, states, and inputs) in the set and *recon_ckt*_1, *recon_ckt*_2, ..., *recon_ckt_B* $\longleftarrow$ rest of the FSMs in the set. *base_ckt*-mode is the default mode of operation for the Reconfigurable FSMIM [6].

The Reconfigurable FSMIM architecture is created by joining the following two parts: (A) Conventional FSMIM architecture [8], & (B) Multiplexer bank (which defines the mode based reconfiguration). The optimal synthesis of the Multiplexer bank is done by iterative greedy heuristic based Hungarian algorithm (IGHA) [6]. At the last phase of IGHA, state transitions of each constituent FSM of the Reconfigurable FSMIM architecture are presented in Figure 1. Therefore, the state encoding of the constituent FSMs altogether affects the LUT requirement of the Reconfigurable FSMIM architecture. At the end of IGHA, a modified description of a single FSM (i.e., *base_ckt*) is obtained which is used to create the Conventional FSMIM part [6].

In FSM implementation on an FPGA platform, state encoding technique acts as a tool for minimizing the hardware consumption [9, 10]. For example, an MCNC FSM benchmark *tbk* requires 82 LUTs when implemented on a Xilinx xc6vlx75t-3 device (Virtex-6) using the Grey encoding technique. But it needs only 41 LUTs on the same platform using the binary encoding technique.

The major problem concerning IGHA is the disintegration of a state encoding technique. It uses binary state encoding as a default state assignment technique for operation [6]. The state assignment method for the Reconfigurable FSMIM architecture leads to an optimization problem as evident from Figure 1. To the best of the authors' knowledge, all the state assignment techniques proposed in the literature provide state codes only for a single FSM.

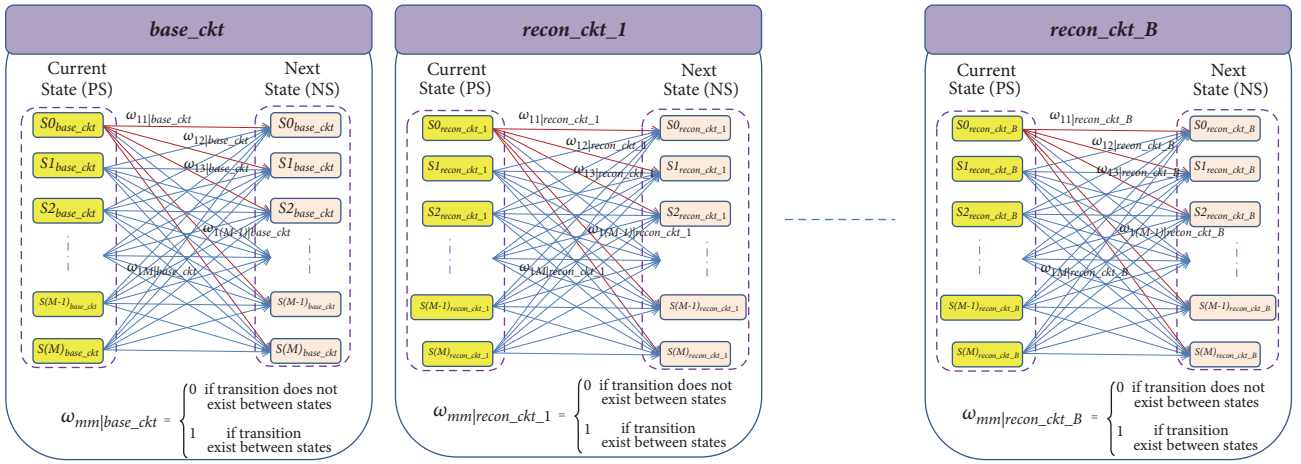

FIGURE 1: State transitions of each constituent FSM of the Reconfigurable FSMIM architecture at the last phase of iterative greedy heuristic based Hungarian algorithm (IGHA).

Therefore, the objective of this work is the integration of IGHA with an optimal state encoding technique to reduce the hardware consumption of Reconfigurable FSMIM on an FPGA platform.

3. Methodology

This work is an extension of work presented in [6]. Hence, all the variables from [6] are used in the same context throughout the article. An improved version of IGHA (Improved-IGHA) is proposed. It addresses the issue of optimal state encoding.

A recent body of literature has investigated the performance of three fundamental types of state encoding techniques on an FPGA platform [9]. The studied methods are as follows: (a) structural approaches, (b) heuristic approaches, and (c) pragmatic approaches. Out of these three approaches, structural state encoding technique outperforms on an FPGA platform [9, 10]. It uses the knowledge of internal structure (i.e., state transition) of the FSM to generate optimal state codes. Therefore, structural information of FSMs is considered to develop the proposed state encoding technique for the Reconfigurable FSMIM.

The structural information of the Reconfigurable FSMIM (i.e., state transition) is obtained from Figure 1. Hence, a unified weight matrix is defined by adding the weight of all component FSMs for the same corresponding states. It is given in (1).

The mathematical formulation of the cost function for an FSM is given in [15]. It uses the structural information (i.e., state transitions) of the particular FSM. Let $\omega_{ij} \longleftarrow$ element of weight matrix and $Dist\_Matrix_{ij}$ be the hamming distance between two particular state codes. $Dist\_Matrix_{ij}$ is obtained by counting the number of 1's after an exclusive-OR operation between the binary state codes as shown in Figure 2. Therefore, from the literature [15], the cost associated with a particular set of state codes (i.e., μ) is defined by (2).

$$
\begin{aligned}
\omega_{11} &= \omega_{11|base\_ckt} + \omega_{11|recon\_ckt\_1} + \omega_{11|recon\_ckt\_2} + \cdots \\
&\quad + \omega_{11|recon\_ckt\_B} \\
&\cdots \\
&\cdots \\
\omega_{1(M-1)} &= \omega_{1(M-1)|base\_ckt} + \omega_{1(M-1)|recon\_ckt\_1} \\
&\quad + \omega_{1(M-1)|recon\_ckt\_2} + \cdots + \omega_{1(M-1)|recon\_ckt\_B} \\
\omega_{1M} &= \omega_{1M|base\_ckt} + \omega_{1M|recon\_ckt\_1} + \omega_{1M|recon\_ckt\_2} \\
&\quad + \cdots + \omega_{1M|recon\_ckt\_B} \\
&\cdots \\
&\cdots \\
\omega_{(M-1)1} &= \omega_{(M-1)1|base\_ckt} + \omega_{(M-1)1|recon\_ckt\_1} \\
&\quad + \omega_{(M-1)1|recon\_ckt\_2} + \cdots + \omega_{(M-1)1|recon\_ckt\_B} \\
\omega_{M1} &= \omega_{M1|base\_ckt} + \omega_{M1|recon\_ckt\_1} + \omega_{M1|recon\_ckt\_2} \\
&\quad + \cdots + \omega_{M1|recon\_ckt\_B} \\
&\cdots \\
&\cdots \\
\omega_{MM} &= \omega_{MM|base\_ckt} + \omega_{MM|recon\_ckt\_1} + \omega_{MM|recon\_ckt\_2} \\
&\quad + \cdots + \omega_{MM|recon\_ckt\_B}
\end{aligned} \tag{1}
$$

$$
cost\,(\mu) = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} \left(\omega_{ij} \times Dist\_Matrix_{ij}\right) \tag{2}
$$

3.1. State Assignment Using Logarithmic Barrier Function Based Gradient Descent Approach for the Reconfigurable FSM. Let the graph described by (2) be $G_{map} = (V_{map}, E_{map})$,

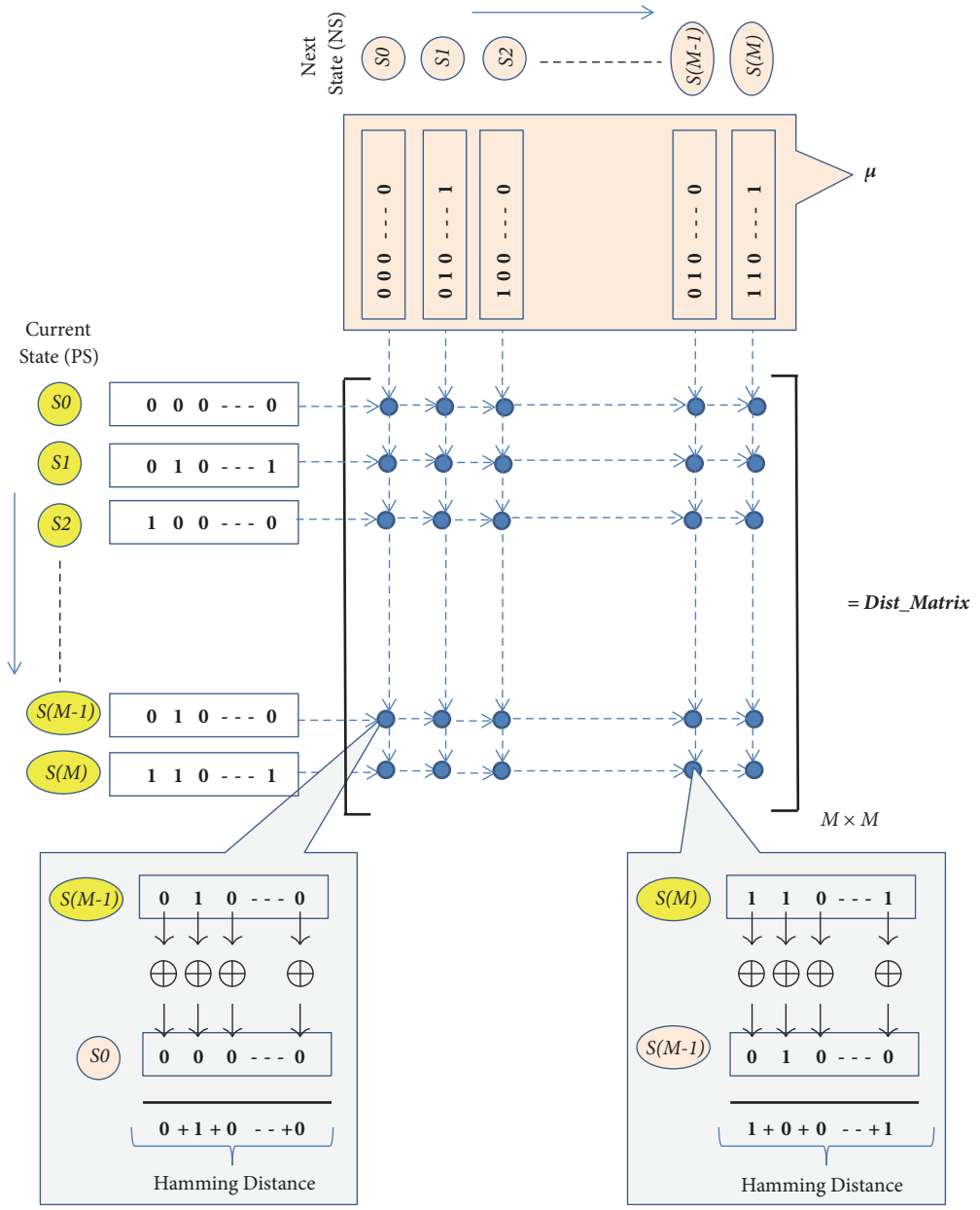

FIGURE 2: Formation of *Dist_Matrix* by calculating the Hamming distance between particular nodes (i.e., states).

where E_{map} (i.e., ω_{ij}) indicates the edge weights between the nodes & V_{map} (i.e., columns of μ) represents the set of nodes. Hence, each node corresponds to a particular binary state code because μ_{ij} opts only the binary labels. M symbolizes the total number of nodes in the graph G_{map}.

Let a hypercube be characterized as $\chi_\eta = (V_\chi, E_\chi)$, where η is the dimension, E_χ is the set of edges, and V_χ is the set of vertices of the hypercube [16]. The cardinality of E_χ and V_χ is given in (3) and (4), respectively.

$$\left|E_\chi\right| = \left|V_\chi\right| \times \frac{\eta}{2} = 2^{\eta-1} \times \eta \tag{3}$$

$$\left|V_\chi\right| = 2^\eta \tag{4}$$

Now, the concept of hypercube embedding is used to reduce (2). An embedding is performed from graph G_{map} onto a hypercube χ_η as described earlier [16, 17]. It is defined as $\mu : V_{map} \longrightarrow V_\chi$ which is a one-to-one mapping function. Consequently, M-binary η-vectors are defined as in (5). Thus, if a node of graph G_{map} (i.e., i) is expressed by a binary state code, the corresponding vertex of the hypercube (i.e., k_i) is represented by the same binary state code.

$$k_i \in \{k : k \in \{0,1\}^\eta\}$$
$$\text{where } i \in V_{map} \ (\textit{i.e., columns of } \mu) \tag{5}$$

In a hypercube, $Dist\_Matrix_{ij}$ $(i, j \in V_{map})$ represents the hamming distance between k_i and k_j. It is shown in

(6), where τ_{ij} is the instantaneous value of k_{ij}. The value of τ_{ij} varies between −1 and 1. Therefore, the cost function is reduced to (7) using hypercube embedding.

$$Dist\_Matrix_{ij} = \sum_{\kappa=1}^{\eta} \left(\tau_{i\kappa} - \tau_{j\kappa}\right)^2$$
$$where,\ k_{ij} = \begin{cases} 1 & if\ \tau_{ij} \geq 0 \\ 0 & if\ \tau_{ij} < 0 \end{cases} \quad (6)$$

$$cost\,(\mu) = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} \sum_{\kappa=1}^{\eta} \left(\omega_{ij} \times \left(\tau_{i\kappa} - \tau_{j\kappa}\right)^2\right) \quad (7)$$

The objective is thus confined to minimize the cost function given in (7). Evidently, it is a discrete optimization problem, where each state can opt only a particular binary state code.

The convergence of Improved-IGHA depends on the convergence of its constituent algorithms, i.e., IGHA and the applied state assignment technique. Therefore, an algorithm with a high convergence speed is preferred to construct the state assignment technique for Improved-IGHA.

The evolutionary technique, such as genetic algorithm (GA), presents a significant shortcoming as its convergence speed slows down near the global optimum [18, 19]. Similarly, particle swarm optimization (PSO) and differential evolution (DE) operate with a high convergence rate but offer premature convergence which is a critical drawback [20, 21]. In the literature, penalty-based approaches, such as Lagrangian technique and logarithmic-barrier function (LBF) method, have proven their potentials to obtain the optimum solution with a high convergence speed [22, 23]. These methods are advantageous in solving a discrete or combinatorial optimization problem [24, 25].

Therefore, the LBF-based Gradient descent approach is adopted to construct the state assignment technique for Improved-IGHA. It is an interior point method that assures the feasible solution. The mathematical formulation of the cost minimization function is performed by LBF. Then, it is reduced iteratively by the gradient-projection approach. The flow chart for the Improved-IGHA is presented in Figure 3.

In LBF technique, the search operation is performed in a continuous space domain to deduce the optimal points. Then, these points are discretized to obtain the optimal solution [26, 27].

In LBF method, an objective function subject to inequality constraints is given in

$$\begin{aligned} \min \quad & f\,(\tau) \\ s.t. \quad & constraint_i\,(\tau) \geq 0, \quad i = 1, \ldots, u \end{aligned} \quad (8)$$

The logarithmic barrier function to minimize the cost function (as in (7)) is given in (9). In LBF search, for any move which omits the constraints, the second term serves as a barrier [28] as shown in

$$\min \psi\,(\tau, \phi) = f\,(\tau) + \phi \sum_{i=1}^{u} \log_e\,(constraint_i\,(\tau)) \quad (9)$$

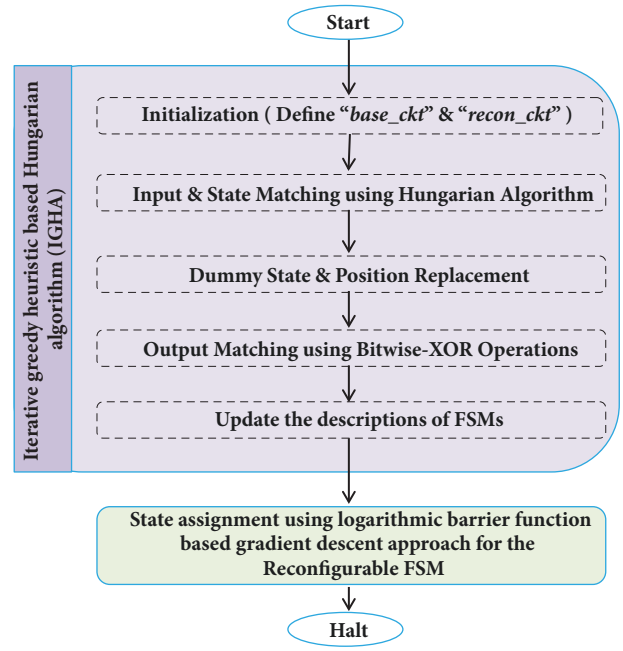

Figure 3: Flow chart for the improved iterative greedy heuristic based Hungarian algorithm (Improved-IGHA).

At the iteration $iter\_t$, (9) is defined as shown in

$$\min \psi\left(\tau, \phi^{iter\_t}\right)_{iter\_t} = f\,(\tau) + \phi^{iter\_t} \sum_{i=1}^{u} \log_e\,(constraint_i\,(\tau)) \quad (10)$$

Initially, LBF selects a feasible τ^0 and $\phi^0 > 0$. Then, it chooses $\phi^{iter\_t+1} = \sigma \cdot \phi^{iter\_t}$, where $\sigma < 1$. This iterative process goes on until ϕ^{iter_t} reaches an adequately small value.

A full-fledged method is required to solve (10) with respect to τ. A first-order gradient-projection approach [29] is well-suited for iteratively minimizing (10). In this approach, the model parameters (a.k.a. weight vectors) are evaluated to minimize the objective function when an analytical calculation is not possible [30, 31]. In this approach, the underlying representation of the objective function of the problem is given in

$$\begin{aligned} \min \quad & \psi\,(\tau, \phi) \\ s.t. \quad & \vartheta\,(\tau) = 0 \end{aligned} \quad (11)$$

An iteration of this projection method is defined by (12). In (12), ρ denotes the step size. ρ is chosen to be a small positive real number [29].

$$\tau \longleftarrow \tau - \rho\left(\nabla \psi\,(\tau, \phi)\right) \quad (12)$$

Thus, small steps (i.e., ρ) are taken in the negative gradient direction of the objective function as illustrated in (12). Then, (13) is used to outline the value of τ on the constraint surface at the next iteration (i.e., $\tau^{(iter\_t+1)}$).

$$\tau \longleftarrow [\tau]^{\vartheta(\tau)=0} \quad (13)$$

The convergence criterion for this iterative process is defined by (14), where $\theta \in [0, 1]$.

$$\left|\tau^{(iter\_t+1)} - \tau^{(iter\_t)}\right| < \theta \tag{14}$$

In this way, embedding problem is reduced to the determination of M-binary η-vectors (as shown in (15)) which optimizes the cost function (i.e., (7)).

$$\begin{aligned} &\tau_i \in \left\{\tau : \tau \in \Re^{\eta} \& \left\|\tau\right\|^2 = 1\right\}; \\ &\qquad i \in V_{map} \text{ where } \left\|\tau\right\|^2 \text{ denotes the norm of } \tau \end{aligned} \tag{15}$$

Hence, the cost function (from (7)) is defined in terms of Hamming distance as shown in

$$\begin{aligned} &cost\left(\mu\right) = \sum_{i=0}^{M-1}\sum_{j=0}^{M-1}\left(\omega_{ij} \times \left\|\tau_i - \tau_j\right\|^2\right) \\ &where,\ \tau_i = \left[\tau_{i1}, \tau_{i2}, \ldots, \tau_{i\eta}\right]^T \ \left\|\tau_i - \tau_j\right\|^2 = \sum_{\kappa=1}^{\eta}\left(\tau_{i\kappa} - \tau_{j\kappa}\right)^2 \end{aligned} \tag{16}$$

The constraint (i.e., boundary condition) for this problem is formed, such as any two vertices on hypercube should not contain the same binary state code (i.e., $\tau_i - \tau_j \neq 0$). Hence, the mathematical representation of the constraint is presented in

$$constraint\left(\tau\right) = \left\|\tau_i - \tau_j\right\|^2 > 0 \tag{17}$$

By applying (16) and (17) on (9), the objective function for LBF is reduced to

$$\begin{aligned} \min \psi\left(\tau, \phi\right) = &\sum_{i=0}^{M-1}\sum_{j=0}^{M-1}\left(\omega_{ij} \times \left\|\tau_i - \tau_j\right\|^2\right) \\ &+ \phi \sum_{i=0}^{M-1}\sum_{j=0}^{M-1}\log_e\left(\left\|\tau_i - \tau_j\right\|^2\right) \end{aligned} \tag{18}$$

Therefore, the entity $\vartheta(\tau)$ (from (13)) is defined by

$$\begin{aligned} &\vartheta\left(\tau\right) = \left[\left(\left\|\tau_1\right\|^2 - 1\right), \left(\left\|\tau_2\right\|^2 - 1\right), \ldots, \left(\left\|\tau_M\right\|^2 - 1\right)\right]^T \\ &s.t. \quad \tau = \left[\tau_1, \tau_2, \ldots, \tau_M\right]^T \\ &\quad \tau_i = \left[\tau_{i1}, \tau_{i2}, \ldots, \tau_{i\eta}\right]^T; \\ &\qquad\qquad where\ each\ \tau_i \ \ (1 \leq i \leq M) \end{aligned} \tag{19}$$

The evaluation of the derivative term (i.e., $\nabla\psi(\tau, \phi)$) is required to move in the gradient descent direction as shown in (12). The needed derivative term is obtained by putting (20), (21), (22), and (23) into (18). Hence, $\nabla\psi(\tau, \phi)$ is defined by (24).

$$\frac{\partial}{\partial\tau_i}\left(\left\|\tau_i - \tau_j\right\|^2\right) = 2\left(\tau_i - \tau_j\right) \tag{20}$$

$$\frac{\partial}{\partial\tau_j}\left(\left\|\tau_i - \tau_j\right\|^2\right) = 2\left(\tau_j - \tau_i\right) \tag{21}$$

$$\frac{\partial}{\partial\tau_i}\left\{\log_e\left(\left\|\tau_i - \tau_j\right\|^2\right)\right\} = \frac{2\left(\tau_i - \tau_j\right)}{\left\|\tau_i - \tau_j\right\|^2} \tag{22}$$

$$\frac{\partial}{\partial\tau_j}\left\{\log_e\left(\left\|\tau_i - \tau_j\right\|^2\right)\right\} = \frac{2\left(\tau_j - \tau_i\right)}{\left\|\tau_i - \tau_j\right\|^2} \tag{23}$$

$$\begin{aligned} &\nabla\psi\left(\tau, \phi\right) \\ &= 2\begin{bmatrix} \sum_{j=1}^{M}\left\{\omega_{1j} \times \left(\tau_1 - \tau_j\right)\right\} + \phi\sum_{j=1}^{M}\left\{\dfrac{\left(\tau_1 - \tau_j\right)}{\left\|\tau_1 - \tau_j\right\|^2}\right\} \\ \sum_{j=1}^{M}\left\{\omega_{2j} \times \left(\tau_2 - \tau_j\right)\right\} + \phi\sum_{j=1}^{M}\left\{\dfrac{\left(\tau_2 - \tau_j\right)}{\left\|\tau_2 - \tau_j\right\|^2}\right\} \\ \cdots \\ \cdots \\ \cdots \\ \sum_{j=1}^{M}\left\{\omega_{Mj} \times \left(\tau_M - \tau_j\right)\right\} + \phi\sum_{j=1}^{M}\left\{\dfrac{\left(\tau_M - \tau_j\right)}{\left\|\tau_M - \tau_j\right\|^2}\right\} \end{bmatrix} \end{aligned} \tag{24}$$

By applying (19) into (13), the normalized vector τ is defined as shown in

$$\left[\tau\right]^{\vartheta(\tau)=0} = \begin{bmatrix} \dfrac{\tau_1}{\left\|\tau_1\right\|} \\ \dfrac{\tau_2}{\left\|\tau_2\right\|} \\ \cdots \\ \cdots \\ \dfrac{\tau_M}{\left\|\tau_M\right\|} \end{bmatrix} \tag{25}$$

If (14) is satisfied, a solution vector which is defined as $\widehat{\tau}_{ij}$ is obtained at the end of the iteration. Therefore, the required set of state codes (i.e., $\widehat{k}_{ij}$) is deduced by discretizing $\widehat{\tau}_{ij}$ using

$$\widehat{k}_{ij} = \begin{cases} 1 & if\ \widehat{\tau}_{ij} \geq 0 \\ 0 & if\ \widehat{\tau}_{ij} < 0 \end{cases} \tag{26}$$

The pseudocode for the proposed state assignment approach is presented in Algorithm 1.

3.2. An Illustrative Example for the Improved Reconfigurable FSMIM Architecture. The following MCNC FSM benchmarks [14] are considered to demonstrate the steps involved in the creation of the Improved Reconfigurable FSMIM architecture:

Input: the objective function defined by Equation (7)
Output: μ^* (i.e., the final state code vector)

begin
Initialization: $\mu \longleftarrow$ *Binary state codes*;
$\phi \longleftarrow$ *initial_*ϕ (*s.t.* $\phi^0 > 0$);
while ($\phi >$ *final_*ϕ) do
repeat
for *iter_t* $\longleftarrow$ 1 *to* θ
$\tau^{iter\_t} \longleftarrow \tau^{(iter\_t-1)} - \rho\{\nabla\psi(\tau, \phi)\}$;
/ * *by Equation* (12)
& *Equation* (24)*/
end
return $\widehat{\tau}_{ij}$ (*i.e. the value of* τ *at the iteration* θ);
evaluate $\widehat{k}_{ij} = \{1, \text{ if } \widehat{\tau}_{ij} \geq 0;\ 0, \text{ if } \widehat{\tau}_{ij} < 0\}$;
/ * *by Equation* (26)*/
Compute cost for the new value of μ *using Equation* (7);
if (*cost*(*old_*μ) ≥ *cost*(*new_*μ)) then *update*, $\mu^* \longleftarrow$ *new_*μ;
else if (*cost*(*old_*μ) < *cost*(*new_*μ)) then *update*, $\mu^* \longleftarrow$ *old_*μ;
end
until *the algorithm converges*
$\phi \longleftarrow \sigma \cdot \phi$;
end
return μ^*;
end

Algorithm 1: State assignment using logarithmic barrier-function based gradient descent approach for the Reconfigurable FSM.

(1) *train*11 (description is provided in Table 1)

(2) *lion*9 (description is provided in Table 2)

The improved Reconfigurable FSMIM architecture is created by joining (A) Conventional FSMIM architecture and (B) Multiplexer bank (which defines the mode based reconfiguration). The optimal synthesis of the Multiplexer bank is done by the proposed Improved-IGHA. At the end of the proposed algorithm, a modified description of a single FSM (i.e., *base_ckt*) is obtained which is used to create the Conventional FSMIM part [6]. The Improved-IGHA consists of the following steps:

(i) **Initialization** (Define *base_ckt* and *recon_ckt*): *train*11 is selected as *base_ckt* because its complexity is greater than *lion*9 as observed from their descriptions. Consequently, *lion*9 acts as *recon_ckt*.

(ii) **Input and State Matching using Hungarian Algorithm**: Input and state matchings are performed together using Algorithms 1, 2, 5, and 6 from [6]. Combinations of input lines of *base_ckt* (i.e., ${}^2P_2 = 2$) are generated. For the first combination ($x_{1|train11}, x_{2|train11}$), states are matched as $S0_{lion9} \longrightarrow S0_{train11}$, $S1_{lion9} \longrightarrow S1_{train11}$, $S3_{lion9} \longrightarrow S2_{train11}$, $S4_{lion9} \longrightarrow S3_{train11}$, $S7_{lion9} \longrightarrow S4_{train11}$, $S2_{lion9} \longrightarrow S5_{train11}$, $S8_{lion9} \longrightarrow S6_{train11}$, $S5_{lion9} \longrightarrow S7_{train11}$, *Dummy state* $\longrightarrow S8_{train11}$, $S6_{lion9} \longrightarrow S9_{train11}$, and *Dummy state* $\longrightarrow S10_{train11}$. It offers zero *assignment_cost* and *total_cost*. For the second combination ($x_{2|train11}, x_{1|train11}$), states are matched as $S0_{lion9} \longrightarrow S0_{train11}$, $S3_{lion9} \longrightarrow S1_{train11}$, $S1_{lion9} \longrightarrow S2_{train11}$, $S4_{lion9} \longrightarrow S3_{train11}$, $S5_{lion9} \longrightarrow S4_{train11}$, $S2_{lion9} \longrightarrow S5_{train11}$, *Dummy state* $\longrightarrow S6_{train11}$, $S7_{lion9} \longrightarrow S7_{train11}$, $S8_{lion9} \longrightarrow S8_{train11}$, $S6_{lion9} \longrightarrow S9_{train11}$, and *Dummy state* $\longrightarrow S10_{train11}$. It also offers zero *assignment_cost* and *total_cost*. Therefore, the first combination ($x_{1|train11}, x_{2|train11}$) is finalized to match with ($x_{1|lion9}, x_{2|lion9}$).

(iii) **Dummy State and Position Replacement**: The replacements of the dummy states and positions in *base_ckt* and *recon_ckt* are performed using Algorithm 3 from [6]. The replaced dummy states (highlighted in "bold italic font") and dummy positions (highlighted in "bold font") are presented in Tables 3 and 4.

(iv) **Output Matching using Bitwise-XOR Operations**: Output Matching is not required in this case, as

Table 1: Description of *train*11 from MCNC FSM Benchmarks [14].

Input		PS	NS	O/P
x_1	x_2			y_1
0	0	S0	S0	0
1	0	S0	S1	-
0	1	S0	S2	-
1	0	S1	S1	1
0	0	S1	S3	1
1	1	S1	S5	1
0	1	S2	S2	1
0	0	S2	S7	1
1	1	S2	S9	1
0	0	S3	S3	1
0	1	S3	S4	1
0	1	S4	S4	1
0	0	S4	S0	-
1	1	S5	S5	1
0	1	S5	S6	1
0	1	S6	S6	1
0	0	S6	S0	-
0	0	S7	S7	1
1	0	S7	S8	1
1	0	S8	S8	1
0	0	S8	S0	-
1	1	S9	S9	1
1	0	S9	S10	1
1	0	S10	S10	1
0	0	S10	S0	-

Table 2: Description of *lion*9 from MCNC FSM Benchmarks [14].

Input		PS	NS	O/P
x_1	x_2			y_1
1	0	S0	S1	0
0	0	S0	S0	0
0	0	S1	S0	0
1	0	S1	S1	0
1	1	S1	S2	0
1	0	S2	S1	0
1	1	S2	S2	0
0	1	S2	S3	0
1	1	S3	S2	1
0	1	S3	S3	1
0	0	S3	S4	1
0	1	S4	S3	1
0	0	S4	S4	1
1	0	S4	S5	1
0	0	S5	S4	1
1	0	S5	S5	1
1	1	S5	S6	1
1	0	S6	S5	1
1	1	S6	S6	1
0	1	S6	S7	1
1	1	S7	S6	1
0	1	S7	S7	1
0	0	S7	S8	1
0	1	S8	S7	1
0	0	S8	S8	1

there is a single output line in *base_ckt* as well as in *recon_ckt*.

(v) **Update the descriptions of FSMs**: The updated descriptions of *base_ckt* and *recon_ckt* are presented in Tables 3 and 4, respectively.

(vi) **State assignment using logarithmic barrier function based gradient descent approach for the Reconfigurable FSM**: The pictorial representation of state transitions for *base_ckt* and *recon_ckt* (from Tables 3 and 4) is given in Figure 4. Therefore, the weight matrix ω is formed using (1). It is given in

$$\omega = \begin{bmatrix} 2 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 1 & 1 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 2 & 1 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (27)$$

The proposed state assignment algorithm starts by considering the binary state codes as an initial solution. It offers the cost as 62 (from (2)).

At the 100^{th} iteration, the instantaneous value τ (from previous iteration, $\tau^{(iter\_99)}$) is obtained as defined by

$$\tau^{(iter\_99)} = \begin{bmatrix} 0.51 & 0.85 & -0.85 & 0.84 \\ -0.52 & -0.85 & -0.51 & 0.85 \\ 0.85 & -0.52 & -0.51 & -0.85 \\ 0.84 & -0.51 & 0.51 & -0.52 \\ -0.51 & 0.51 & 0.85 & -0.52 \\ -0.85 & -0.52 & -0.85 & -0.51 \\ -0.52 & 0.84 & -0.51 & -0.85 \\ 0.84 & 0.51 & -0.52 & -0.85 \\ 0.51 & -0.85 & -0.85 & 0.51 \\ 0.85 & 0.84 & 0.51 & -0.52 \\ -0.51 & 0.51 & -0.52 & 0.85 \end{bmatrix}^T \quad (28)$$

Table 3: Updated description of *train*11.

Input		PS	NS	O/P
x_1	x_2			y_1
0	0	S0	S0	0
1	0	S0	S1	-
0	1	S0	S2	-
1	0	S1	S1	1
0	0	S1	S3	1
1	1	S1	S5	1
0	1	S2	S2	1
0	0	S2	S7	1
1	1	S2	S9	1
0	0	S3	S3	1
0	1	S3	S4	1
0	**0**	**S3**	**S3**	**1**
0	1	S4	S4	1
0	0	S4	S0	-
0	**1**	**S4**	**S4**	**1**
1	1	S5	S5	1
0	1	S5	S6	1
1	**1**	**S5**	**S5**	**1**
0	1	S6	S6	1
0	0	S6	S0	-
0	0	S7	S7	1
1	0	S7	S8	1
1	**0**	**S7**	**S8**	**1**
1	0	S8	S8	1
0	0	S8	S0	-
0	**0**	**S8**	**S0**	**-**
1	1	S9	S9	1
1	0	S9	S10	1
1	**1**	**S9**	**S9**	**1**
1	0	S10	S10	1
0	0	S10	S0	-
0	**0**	**S10**	**S0**	**-**

Table 4: Updated description of *lion*9.

Input		PS	NS	O/P
x_1	x_2			y_1
0	0	S0	S0	0
1	0	S0	S1	0
0	**0**	**S0**	**S0**	**0**
1	0	S1	S1	0
0	0	S1	S0	0
1	1	S1	S5	0
0	1	S2	S2	1
0	0	S2	S3	1
1	1	S2	S5	1
0	0	S3	S3	1
0	1	S3	S2	1
1	0	S3	S7	1
0	1	S4	S4	1
0	0	S4	S6	1
1	1	S4	S9	1
1	1	S5	S5	0
0	1	S5	S2	0
1	0	S5	S1	0
0	1	S6	S4	1
0	0	S6	S6	1
0	0	S7	S3	1
1	0	S7	S7	1
1	1	S7	S9	1
1	***0***	***S8***	***S1***	***0***
0	***0***	***S8***	***S0***	***0***
0	***0***	***S8***	***S0***	***0***
1	1	S9	S9	1
1	0	S9	S7	1
0	1	S9	S4	1
1	***0***	***S10***	***S1***	***0***
0	***0***	***S10***	***S0***	***0***
0	***0***	***S10***	***S0***	***0***

The derivative (from (24)) is evaluated as defined by

$$\nabla\psi(\tau,\phi) = \begin{bmatrix} -1517 & -1224 & -2118 & 2126 \\ -1739 & 1258 & -2622 & 2960 \\ 2135 & 701 & 1558 & 402 \\ -1223 & 1598 & 1793 & 736 \\ 722 & -739 & 2886 & 760 \\ 1262 & 723 & -2888 & 1550 \\ 1506 & -1209 & -1775 & 401 \\ -398 & 1754 & 763 & -2889 \\ -1549 & -2118 & 418 & -1579 \\ 2945 & 2953 & 1761 & -1736 \\ -2583 & -1504 & 1506 & 2079 \end{bmatrix}^T \quad (29)$$

So, the current value of τ (i.e., $\tau^{(iter\_100)}$) is obtained from (12). It is given in (30) by choosing $\rho = 10^{-3}$ (a very small value).

$$\tau^{(iter\_100)} = \begin{bmatrix} 2.027 & 2.074 & 1.268 & -1.286 \\ 1.219 & -2.108 & 2.112 & -2.11 \\ -1.285 & -1.221 & -2.068 & -1.252 \\ 2.063 & -2.108 & -1.283 & -1.256 \\ -1.232 & 1.249 & -2.036 & -1.28 \\ -2.112 & -1.243 & 2.038 & -2.06 \\ -2.026 & 2.049 & 1.265 & -1.251 \\ 1.238 & -1.244 & -1.283 & 2.039 \\ 2.059 & 1.268 & -1.268 & 2.089 \\ -2.095 & -2.113 & -1.251 & 1.216 \\ 2.073 & 2.014 & -2.026 & -1.229 \end{bmatrix}^T \quad (30)$$

Then, τ is directed towards the unity radius hypersphere. It is given in

$$\widehat{\tau} = \begin{bmatrix} 0.85 & 0.84 & 0.51 & -0.52 \\ 0.51 & -0.85 & 0.85 & -0.85 \\ -0.52 & -0.51 & -0.85 & -0.51 \\ 0.84 & -0.85 & -0.52 & -0.51 \\ -0.51 & 0.51 & -0.85 & -0.52 \\ -0.85 & -0.52 & 0.84 & -0.85 \\ -0.85 & 0.85 & 0.51 & -0.51 \\ 0.51 & -0.51 & -0.52 & 0.85 \\ 0.85 & 0.51 & -0.51 & 0.85 \\ -0.85 & -0.85 & -0.51 & 0.51 \\ 0.84 & 0.84 & -0.85 & -0.51 \end{bmatrix}^{T} \begin{matrix} \longleftarrow S0 \\ \longleftarrow S1 \\ \longleftarrow S2 \\ \longleftarrow S3 \\ \longleftarrow S4 \\ \longleftarrow S5 \\ \longleftarrow S6 \\ \longleftarrow S7 \\ \longleftarrow S8 \\ \longleftarrow S9 \\ \longleftarrow S10 \end{matrix} \quad (31)$$

The required set of state codes is deduced as $S0 \longrightarrow 1110$, $S1 \longrightarrow 1010$, $S2 \longrightarrow 0000$, $S3 \longrightarrow 1000$, $S4 \longrightarrow 0100$, $S5 \longrightarrow 0010$, $S6 \longrightarrow 0110$, $S7 \longrightarrow 1001$, $S8 \longrightarrow 1101$, $S9 \longrightarrow 0001$, and $S10 \longrightarrow 1100$ by discretizing the current value of τ using (26). Hence, the cost is reduced to 48 (from (2)).

In the end, a Bitwise-XOR operation is performed between the updated descriptions of *train*11 and *lion*9. It provides the Multiplexer bank (i.e., part-B). The updated descriptions of *train*11 are used to construct the Conventional FSMIM part (i.e., part-A).

4. Numerical Results and Discussions

To validate the proposed approach, experiments have been performed using MCNC FSM benchmarks [14]. MATLAB (2016b) environment is used to implement the proposed Improved-IGHA. It produces the optimized description for the constituting parts of the Improved Reconfigurable FSMIM architecture. The obtained description is then converted into the Verilog HDL code using MATLAB HDL Coder tool-box. The implementation of the Improved Reconfigurable FSMIM architecture is performed on the Virtex-6 speed-3 device as in [6, 11]. The configuration of the workstation to execute computations is as follows: Intel(R) Core i7 (6th Gen), 16 GB RAM, and 3.5 GHz CPU.

In Improved-IGHA, combinations of input lines, states, and output lines are generated using permutation to perform input, state, and output matching, respectively. The number of input and output lines used for matching is restricted to 7 (i.e., ${}^{7}P_{7} = 5040$ combinations) to utilize the resources efficiently. Hence, the information content of an input/output line becomes the criteria for selection. An input/output line with high information content is preferred.

The following MCNC FSM benchmarks [14] are selected to illustrate the implementation of the Improved Reconfigurable FSMIM architecture and present its comparative analysis with the existing literature: *s*1494, *s*832, *s*208, *planet*, *s*386, *sand*, *mc*, *styr*, *cse*, *ex*6, *planet*1, and *s*1488.

*s*1494 is chosen as *base_ckt* (i.e., the circuit added at the 0^{th} iteration of Improved-IGHA), as it is more complex (i.e., the total number of transitions is high) as compared with the other FSMs in the set. The other FSMs in the set are added iteratively in the design in their respective order.

In an FSM, a specific state is chosen only if a particular set of input bits (i.e., 1's or 0's) are present. Hence, the percentage of 1's and 0's together in an input line acts as information content as shown in Table 5 (the selected input lines to match with *base_ckt* are highlighted). Similarly, the output is always defined by "1." Hence, the percentage of 1's in an output line serves as information content as shown in Tables 6 and 7 (the selected output lines to match with *base_ckt* are highlighted).

At the first phase of Improved-IGHA, input and state matching are performed together, and optimal assignments (with respect to *base_ckt*) are made. It is presented in Table 5. All the *recon_ckt* states are mapped onto *base_ckt* states in their respective order. Output matching (with respect to *base_ckt*) is performed iteratively by Bitwise-XOR operations. It is presented in Tables 6 and 7. Then, after updating the descriptions of constituting FSMs, the state assignment using logarithmic barrier function based gradient descent approach is performed.

To present a comparative analysis of the total computation time required by IGHA [6] and Improved-IGHA, an inbuilt feature in MATLAB named "stopwatch timer" is used. It evaluates the elapsed time (i.e., the execution time between the starting and stopping of a function). As evident from the literature [6], linear assignment problems (LAPs) are solved several times by IGHA to perform matchings among all generated combinations to add $recon\_ckt\_b \in \{recon\_ckt\_1, \ldots, recon\_ckt\_B\}$ iteratively. The convergence period of IGHA to solve a single LAP ranges from 0.03 ms to 0.6 ms. Hence, the total elapsed time taken by IGHA (i.e., t_{IG}) is given in (32). The convergence time for the state assignment using LBF-based gradient descent approach (i.e., t_{SE}) to add $recon\_ckt\_b \in \{recon\_ckt\_1, \ldots, recon\_ckt\_B\}$ iteratively is given in Table 8. Therefore, the total elapsed time taken by Improved-IGHA (i.e., $t_{Proposed}$) is an addition of t_{IG} and t_{SE} (from Figure 3). It is presented in Table 8.

$$\begin{aligned} &\textit{Total elapsed time for IGHA} \\ &= \sum_{recon\_ckt\_b} (1+E)\left(M\_base \times M\_recon\right) \\ &\times \textit{ Elapsed time for IGHA to solve a single LAP} \\ &s.t.\ E = \begin{cases} {}^{L\_base}P_{L\_recon} & \text{if } L\_base \geq L\_recon; \\ {}^{L\_recon}P_{L\_base} & \text{if } L\_base < L\_recon; \end{cases} \\ &recon\_ckt\_b \in \{recon\_ckt\_1, \ldots, recon\_ckt\_B\}. \end{aligned} \quad (32)$$

The experimental results presented in Table 8 illustrate that the total computation time required by IGHA is far higher than the convergence time for the proposed state assignment technique (i.e., $t_{IG} \gg t_{SE}$). Therefore, the total computation time required by Improved-IGHA is equivalent to the total computation time needed by IGHA (i.e., $t_{Proposed} \cong t_{IG}$).

Table 5: The information content for input lines of MCNC FSM Benchmarks and their matching with input lines of *base_ckt*.

FSM	No. of I/P	Input lines with their information content	Matched with *base_ckt*	No. of state
*s*1494	8	$\mathbf{x_1}$(**99.6%**), x_2(10%), $\mathbf{x_3}$(**25.6%**), $\mathbf{x_4}$(**24.8%**), $\mathbf{x_5}$(**12.4%**), $\mathbf{x_6}$(**66%**), $\mathbf{x_7}$(**41.6%**), $\mathbf{x_8}$(**20.4%**)	x_1, x_3, x_4, x_5, x_6, x_7, x_8	48
*s*832	18	x_1(2.04%), x_2(6.93%), x_3(2.44%), x_4(4.48%), $\mathbf{x_5}$(**70.61%**), x_6(1.63%), $\mathbf{x_7}$(**14.28%**), $\mathbf{x_8}$(**9.79%**), $\mathbf{x_9}$(**8.97%**), x_{10}(8.16%), $\mathbf{x_{11}}$(**8.57%**), x_{12}(6.12%), x_{13}(4.081%), x_{14}(4.08%), x_{15}(1.63%), x_{16}(1.63%), $\mathbf{x_{17}}$(**59.18%**), $\mathbf{x_{18}}$(**81.63%**),	x_9, x_{11}, x_{18}, x_{17}, x_8, x_7, x_5	25
*s*208	11	$\mathbf{x_1}$(**96.73%**), $\mathbf{x_2}$(**91.5%**), x_3(0%), x_4(0%), x_5(0%), $\mathbf{x_6}$(**2.61%**), x_7(2.61%), $\mathbf{x_8}$(**5.22%**), $\mathbf{x_9}$(**12.41%**), $\mathbf{x_{10}}$(**49.01%**), $\mathbf{x_{11}}$(**77.12%**)	x_1, x_8, x_6, x_9, x_{11}, x_2, x_{10}	18
planet	7	$\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_5}$, $\mathbf{x_6}$, $\mathbf{x_7}$	x_6, x_3, x_4, x_2, x_5, x_7, x_1	48
*s*386	7	$\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_5}$, $\mathbf{x_6}$, $\mathbf{x_7}$	x_4, x_3, x_2, x_7, x_1, x_5, x_6	13
sand	11	$\mathbf{x_1}$(**52.17%**), $\mathbf{x_2}$(**52.17%**), $\mathbf{x_3}$(**52.17%**), $\mathbf{x_4}$(**52.17%**), $\mathbf{x_5}$(**24.45%**), x_6(3.26%), x_7(18.47%), $\mathbf{x_8}$(**26.63%**), x_9(1.08%), $\mathbf{x_{10}}$(**79.89%**), x_{11}(19.56%)	x_4, x_2, x_1, x_{10}, x_5, x_3, x_8	32
mc	3	$\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$	−, x_3, x_1,−, x_2, −, −	4
styr	9	$\mathbf{x_1}$(**92.16%**), $\mathbf{x_2}$(**4.81%**), $\mathbf{x_3}$(**48.79%**), $\mathbf{x_4}$(**69.87%**), $\mathbf{x_5}$(**68.67%**), $\mathbf{x_6}$(**39.15%**), x_7(4.81%), $\mathbf{x_8}$(**5.42%**), x_9(3.61%)	x_2, x_4, x_5, x_3, x_8, x_6, x_1	30
cse	7	$\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_5}$, $\mathbf{x_6}$, $\mathbf{x_7}$	x_5, x_2, x_7, x_3, x_4, x_1, x_6	16
*ex*6	5	$\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_5}$	x_3, x_4, −, x_1, x_2, −, x_5	8

Table 5: Continued.

FSM	No. of I/P	Input lines with their information content	Matched with *base_ckt*	No. of state
*planet*1	7	$\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$, $\mathbf{x_4}$, $\mathbf{x_5}$, $\mathbf{x_6}$, $\mathbf{x_7}$	x_5, x_1, x_6, x_3, x_4, x_2, x_7	48
*s*1488	8	$\mathbf{x_1}$(**98.8%**), x_2(12.74%), $\mathbf{x_3}$(**23.5%**), $\mathbf{x_4}$(**23.9%**), $\mathbf{x_5}$(**16.33%**), $\mathbf{x_6}$(**65.73%**), $\mathbf{x_7}$(**40.23%**), $\mathbf{x_8}$(**18.32%**)	x_1, x_5, x_3, x_4, x_8, x_7, x_6	48

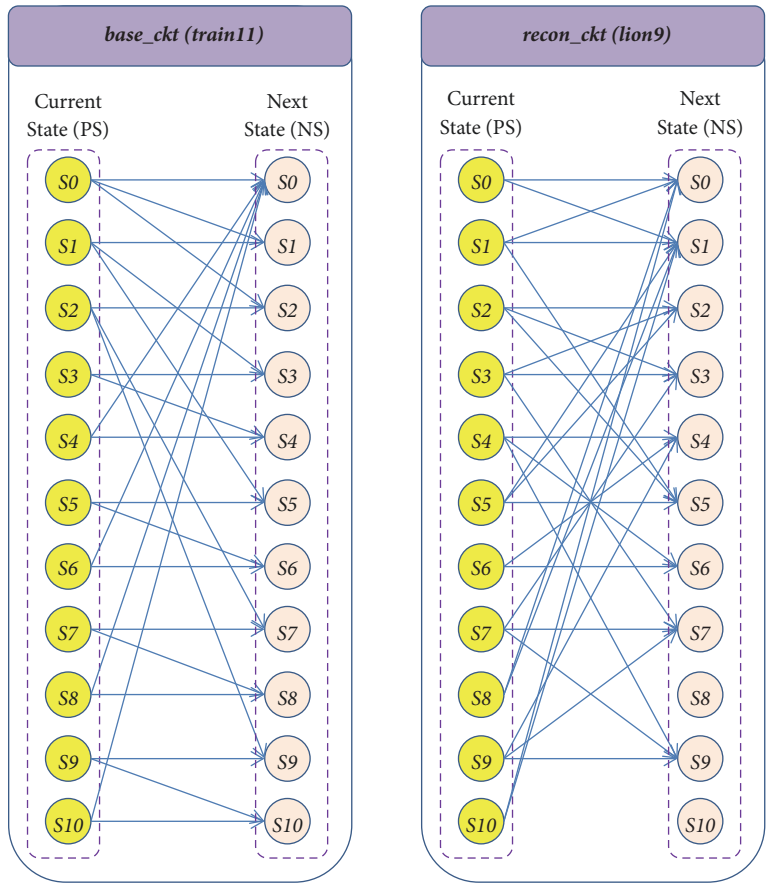

Figure 4: The pictorial representation of state transitions for *base_ckt* and *recon_ckt*.

Convergence plot for the state assignment using logarithmic barrier function based gradient descent approach after adding the last constituting FSM in the proposed architecture is presented in Figure 5. It starts by taking binary state codes as an initial code. The cost offered to the proposed architecture is calculated by (2). It converges to 200 iterations. The cost is reduced from 1028 to 923 as shown in Figure 5. Consequently, at 200^{th} iteration, the following state codes are obtained: $S0 \longrightarrow 010111$, $S1 \longrightarrow 000000$, $S2 \longrightarrow 000111$, $S3 \longrightarrow 110000$, $S4 \longrightarrow 010100$, $S5 \longrightarrow 110101$, $S6 \longrightarrow 000110$, $S7 \longrightarrow 011101$, $S8 \longrightarrow 001100$, $S9 \longrightarrow 011010$, $S10 \longrightarrow 011110$, $S11 \longrightarrow 001110$, $S12 \longrightarrow 010110$, $S13 \longrightarrow 111011$, $S14 \longrightarrow 000011$, $S15 \longrightarrow 100110$, $S16 \longrightarrow 110111$, $S17 \longrightarrow 001010$, $S18 \longrightarrow 011100$, $S19 \longrightarrow 100001$, $S20 \longrightarrow 101001$, $S21 \longrightarrow 110010$, $S22 \longrightarrow 100000$, $S23 \longrightarrow 001000$, $S24 \longrightarrow 001111$, $S25 \longrightarrow 101101$, $S26 \longrightarrow 010011$, $S27 \longrightarrow 101010$, $S28 \longrightarrow 110001$, $S29 \longrightarrow 001011$, $S30 \longrightarrow 111010$, $S31 \longrightarrow 011111$, $S32 \longrightarrow 000101$, $S33 \longrightarrow 000100$, $S34 \longrightarrow 111101$, $S35 \longrightarrow 000001$, $S36 \longrightarrow 001101$, $S37 \longrightarrow 101000$, $S38 \longrightarrow 000010$, $S39 \longrightarrow 100111$, $S40 \longrightarrow 110110$, $S41 \longrightarrow 011001$, $S42 \longrightarrow 100010$, $S43 \longrightarrow 001001$, $S44 \longrightarrow 010001$, $S45 \longrightarrow 100101$, $S46 \longrightarrow 101111$, and $S47 \longrightarrow 010010$.

Table 6: The information content for output lines of MCNC FSM Benchmarks & their matching with output lines of *base_ckt*.

FSM	No. of O/P	Output lines with their information content	Matched with *base_ckt*
*s*1494	19	y_1(24.8%), y_2(4.8%), y_3(5.2%), y_4(3.2%), y_5(2.4%), y_6(2.4%), y_7(15.2%), y_8(25.2%), y_9(1.6%), y_{10}(6.4%), $\mathbf{y_{11}}$**(87.2%)**, $\mathbf{y_{12}}$**(40.4%)**, $\mathbf{y_{13}}$**(32.8%)**, $\mathbf{y_{14}}$**(70.4%)**, $\mathbf{y_{15}}$**(38.4%)**, y_{16}(18.4%), $\mathbf{y_{17}}$**(70%)**, y_{18}(31.2%), $\mathbf{y_{19}}$**(49.2%)**	y_{11}, y_{12}, y_{13}, y_{14}, y_{15}, y_{17}, y_{19}
*s*832	19	$\mathbf{y_1}$**(5.71%)**, $\mathbf{y_2}$**(2.44%)**, y_3(1.22%), y_4(1.63%), y_5(2.44%), y_6(0.81%), $\mathbf{y_7}$**(2.44%)**, $\mathbf{y_8}$**(73.06%)**, y_9(0.81%), y_{10}(0.81%), $\mathbf{y_{11}}$**(5.3%)**, y_{12}(2.44%), y_{13}(0.81%), y_{14}(1.63%), $\mathbf{y_{15}}$**(6.12%)**, y_{16}(0.81%), y_{17}(1.63%), y_{18}(2.44%), $\mathbf{y_{19}}$**(41.22%)**	y_{19}, y_{15}, y_2, y_8, y_7, y_1, y_{11}
*s*208	2	$\mathbf{y_1}$, $\mathbf{y_2}$	–, y_1, –, –, y_2, –, –
planet	19	$\mathbf{y_1}$**(54.78%)**, y_2(23.47%), $\mathbf{y_3}$**(69.56%)**, y_4(16.52%), $\mathbf{y_5}$**(32.17%)**, $\mathbf{y_6}$**(73.91%)**, $\mathbf{y_7}$**(26.08%)**, $\mathbf{y_8}$**(28.69%)**, $\mathbf{y_9}$**(91.3%)**, y_{10}(4.34%), y_{11}(1.73%), y_{12}(22.6%), y_{13}(11.3%), y_{14}(2.6%), y_{15}(3.47%), y_{16}(1.73%), y_{17}(3.47%), y_{18}(3.47%), y_{19}(20%)	y_6, y_8, y_9, y_5, y_3, y_1, y_7
*s*386	7	$\mathbf{y_1}$, $\mathbf{y_2}$, $\mathbf{y_3}$, $\mathbf{y_4}$, $\mathbf{y_5}$, $\mathbf{y_6}$,$\mathbf{y_7}$	y_6, y_1, y_3, y_4, y_7, y_5, y_2
sand	9	$\mathbf{y_1}$**(22.28%)**, $\mathbf{y_2}$**(36.41%)**, $\mathbf{y_3}$**(16.84%)**, $\mathbf{y_4}$**(62.5%)**, $\mathbf{y_5}$**(15.76%)**, $\mathbf{y_6}$**(8.15%)**, $\mathbf{y_7}$**(17.39%)**, y_8(1.63%), y_9(3.26%)	y_4, y_6, y_2, y_7, y_3, y_5, y_1
mc	5	$\mathbf{y_1}$, $\mathbf{y_2}$, $\mathbf{y_3}$, $\mathbf{y_4}$, $\mathbf{y_5}$	y_4, y_2, y_5, y_1, y_3, –, –
styr	10	$\mathbf{y_1}$**(15.66%)**, $\mathbf{y_2}$**(33.73%)**, $\mathbf{y_3}$**(25.9%)**, y_4(3.012%), $\mathbf{y_5}$**(8.43%)**, $\mathbf{y_6}$**(7.22%)**, $\mathbf{y_7}$**(5.42%)**, $\mathbf{y_8}$**(11.445%)**, y_9(3.614%), y_{10}(4.819%)	y_5, y_6, y_7, y_8, y_3, y_2, y_1
cse	7	$\mathbf{y_1}$, $\mathbf{y_2}$, $\mathbf{y_3}$, $\mathbf{y_4}$, $\mathbf{y_5}$, $\mathbf{y_6}$, $\mathbf{y_7}$	y_7, y_3, y_5, y_1, y_4, y_6, y_2
*ex*6	8	$\mathbf{y_1}$**(58.82%)**, $\mathbf{y_2}$**(29.41%)**, $\mathbf{y_3}$**(55.88%)**, $\mathbf{y_4}$**(29.41%)**, $\mathbf{y_5}$**(50%)**, $\mathbf{y_6}$**(26.47%)**, y_7(5.88%), $\mathbf{y_8}$**(23.52%)**	y_5, y_8, y_4, y_1, y_6, y_2, y_3

Table 7: The information content for output lines of MCNC FSM Benchmarks and their matching with output lines of *base_ckt*.

FSM	No. of O/P	Output lines with their information content	Matched with *base_ckt*
*planet*1	19	$\mathbf{y_1}$**(54.78%)**, y_2(23.47%), $\mathbf{y_3}$**(69.56%)**, y_4(16.52%), $\mathbf{y_5}$**(32.17%)**, $\mathbf{y_6}$**(73.91%)**, $\mathbf{y_7}$**(26.08%)**, $\mathbf{y_8}$**(28.69%)**, $\mathbf{y_9}$**(91.3%)**, y_{10}(4.34%), y_{11}(1.73%), y_{12}(22.6%), y_{13}(11.3%), y_{14}(2.6%), y_{15}(3.47%), y_{16}(1.73%), y_{17}(3.47%), y_{18}(3.47%), y_{19}(20%)	y_6, y_7, y_1, y_8, y_5, y_3, y_9
*s*1488	19	y_1(2.39%), y_2(2.78%), y_3(1.59%), y_4(5.17%), y_5(2.39%), y_6(15.13%), $\mathbf{y_7}$**(71.31%)**, y_8(3.98%), $\mathbf{y_9}$**(51.39%)**, y_{10}(6.3%), y_{11}(16.7%), $\mathbf{y_{12}}$**(37.1%)**, $\mathbf{y_{13}}$**(70.5%)**, y_{14}(24.3%), $\mathbf{y_{15}}$**(87.6%)**, y_{16}(31.1%), y_{17}(25.4%), $\mathbf{y_{18}}$**(31.4%)**, $\mathbf{y_{19}}$**(39.84%)**	y_{18}, y_7, y_{13}, y_{19}, y_9, y_{15}, y_{12}

Table 8: Comparative analysis of the total computation time required by IGHA [6] and Improved-IGHA.

Iteration No.	FSM included in the specific iteration	Total elapsed time for IGHA [6] (Hrs) t_{IG}	Total elapsed time for state encoding tech. (ms) t_{SE}	Total elapsed time for Improved-IGHA (Hrs) $t_{Proposed} = t_{IG} + t_{SE}$ $\because t_{IG} \gg t_{SE}$, $\therefore t_{Proposed} \cong t_{IG}$
0^{th}	*s*1494	0	296.529	0
1^{st}	*s*832	25.34	214.468	25.34
2^{nd}	*s*208	18.57	156.062	18.57
3^{rd}	*planet*	48.65	338.560	48.65
4^{th}	*s*386	13.17	182.808	13.17
5^{th}	*sand*	32.43	293.894	32.43
6^{th}	*mc*	0.182	148.784	0.182
7^{th}	*styr*	30.409	249.509	30.409
8^{th}	*cse*	16.21	387.923	16.21
9^{th}	*ex*6	4.38	167.144	4.38
10^{th}	*planet*1	48.544	312.809	48.544
11^{th}	*s*1488	48.654	326.406	48.654

At the last phase of Improved-IGHA, a mutual Bitwise-XOR operation is conducted between the updated descriptions of FSMs. Therefore, the constituting parts of the proposed architecture are created. The individual share of constituent FSMs in the Improved-Reconfigurable FSMIM architecture is determined by the difference between the occupied LUTs in the recent and its previous iteration. After adding all the constituting FSMs in the proposed design (i.e., at the last iteration), the total LUT consumption and operating frequency are obtained. It is presented in Table 9.

Experimental results for the proposed architecture illustrates a significant area reduction by an average of 20.38% and speed improvement by an average of 32.73% over VRMUX [11] during FPGA implementation. It also demonstrates an adequate area reduction by an average of 16.05% and speed improvement by an average of 1.77% over Reconfigurable FSMIM-S architecture [6] during FPGA implementation. When these results are compared with CRMUX [11], a speed improvement by an average of 11.06% is obtained. The proposed architecture requires an average of 58.38% more LUTs as compared with CRMUX [11] during FPGA implementation. It is the only trade-off for the proposed design. A comparative analysis of the hardware consumption and maximum operating frequency variation on FPGA implementation is presented in Figures 6 and 7, respectively.

5. Concluding Remarks

This article furnishes the framework for the Improved-Reconfigurable FSMIM architecture. The Improved-Reconfigurable FSMIM architecture is created by joining the following two parts: (A) Conventional FSMIM architecture and (B) Multiplexer bank (which defines the mode based reconfiguration). An improved version of iterative greedy

Table 9: Implementation of the Improved Reconfigurable FSMIM architecture on the Virtex-6 speed-3 device in an iterative manner.

Iteration No.	FSM included in the specific iteration	#LUTs occupied in the specific iteration	Maximum Operating Frequency (MHz)	Maximum Path Delay (ns)	#LUTs occupied by the FSM (#LUTs in the current iteration - #LUTs in the previous iteration)
0^{th}	*s*1494	40	831.693	3.898	40
1^{st}	*s*832	97	803.44	4.288	57
2^{nd}	*s*208	114	793.583	5.252	17
3^{rd}	*planet*	142	785.326	4.219	28
4^{th}	*s*386	157	776.863	4.534	15
5^{th}	*sand*	187	760.88	4.117	30
6^{th}	*mc*	198	757.237	3.854	11
7^{th}	*styr*	217	743.431	4.204	19
8^{th}	*cse*	240	713.929	4.401	23
9^{th}	*ex*6	249	704.892	4.649	9
10^{th}	*planet*1	274	690.83	4.977	25
11^{th}	*s*1488	293	676.928	5.151	19

#LUTs ⟶ number of LUTs occupied in ISE

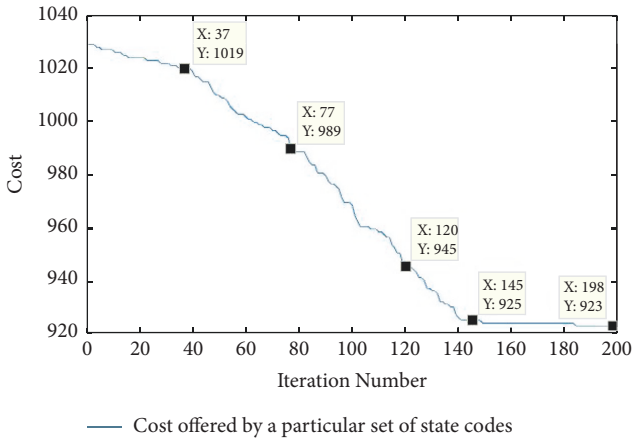

Figure 5: Convergence plot for the state assignment using logarithmic barrier function based gradient descent approach after adding the last constituting FSM in the proposed architecture.

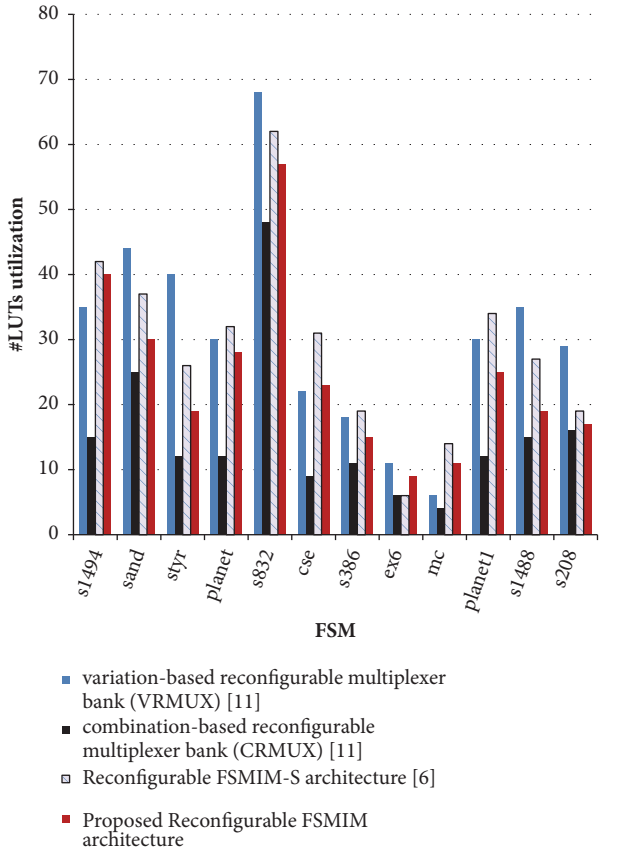

Figure 6: Comparative analysis of the number of LUT consumption on FPGA implementation.

heuristic based Hungarian algorithm (Improved-IGHA) is proposed to establish the constituting parts as mentioned earlier. Improved-IGHA is an integration of IGHA [6] and a state assignment using logarithmic barrier function based gradient descent approach. It reduces the hardware consumption of the proposed architecture by performing an optimal state encoding. An illustrative example using MCNC FSM benchmarks is also given to demonstrate the steps involved in the creation of the proposed architecture.

The proposed architecture illustrates a significant area reduction by an average of 20.38% and speed improvement by an average of 32.73% over VRMUX [11] during FPGA implementation. It also demonstrates an adequate area reduction by an average of 16.05% and speed improvement by an average of 1.77% over Reconfigurable FSMIM-S architecture [6] during FPGA implementation. When these results are compared with CRMUX [11], a speed improvement by an average of 11.06% is obtained. The proposed architecture requires an average of 58.38% more LUTs as compared with CRMUX [11] during FPGA implementation. It is the only trade-off for the proposed design.

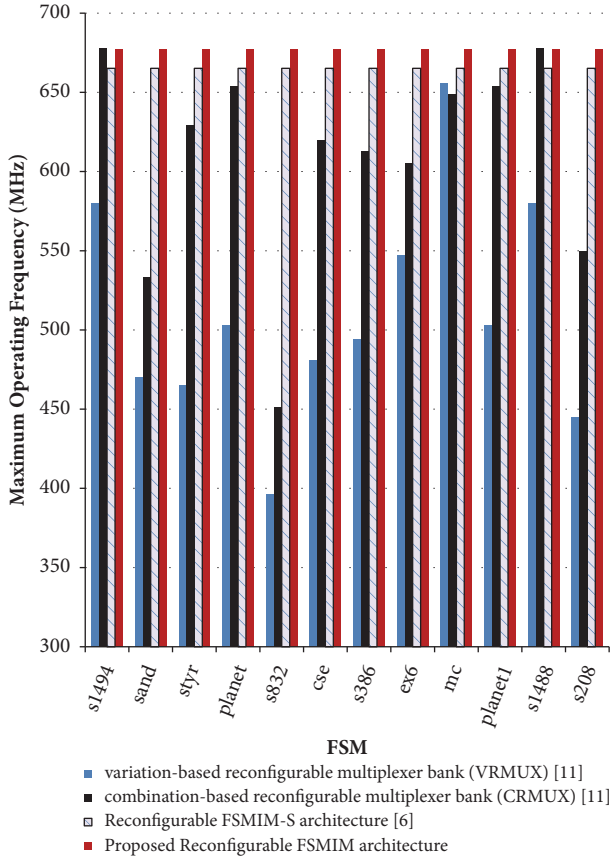

Figure 7: Comparative analysis of maximum operating frequency on FPGA implementation.

Further, the proposed architecture will be investigated to develop an efficient architecture for multistage signal processing [1, 2] and circuit testing [5] based applications.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The datasets generated during and/or analyzed during the current study are available in [6] repository [DOI: 10.1155/2018/6831901]. This work is conducted in the Department of ECE, SRM Institute of Science and Technology, Kattankulathur-603203, Chennai, India.

References

[1] E. De Lucas, M. Sanchez-Elez, and I. Pardines, "DSPONE48: a methodology for automatically synthesize HDL focus on the reuse of DSP slices," *Journal of Parallel and Distributed Computing*, vol. 106, pp. 132–142, 2017.

[2] J. Wu, D. Yang, and Z. Chen, "Design and application of multi-stage reconfigurable signal processing flow on FPGA," *Computers and Electrical Engineering*, vol. 42, pp. 1–11, 2015.

[3] J. Zheng, W. Gao, D. Wu, and D. Xie, "An efficient VLSI architecture for CBAC of AVS HDTV decoder," *Signal Processing: Image Communication*, vol. 24, no. 4, pp. 324–332, 2009.

[4] N. I. Rafla and I. Gauba, "A reconfigurable pattern matching hardware implementation using on-chip RAM-based FSM," in *Proceedings of the 53rd IEEE International Midwest Symposium on Circuits and Systems, MWSCAS 2010*, pp. 49–52, IEEE, Seattle, Wash, USA, August 2010.

[5] Z. Ling, K. Ji-Shun, and Y. Zhi-Qiang, "Virtual scan chains reordering using a RAM-based module for high test compression," *Microelectronics Journal*, vol. 43, no. 11, pp. 869–872, 2012.

[6] N. Das and P. A. Priya, "FPGA implementation of reconfigurable finite state machine with input multiplexing architecture using hungarian method," *International Journal of Reconfigurable Computing*, vol. 2018, Article ID 6831901, 15 pages, 2018.

[7] J. Glaser, M. Damm, J. Haase, and C. Grimm, "TR-FSM: transition-based reconfigurable finite state machine," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 4, no. 3, article no. 23, pp. 1–14, 2011.

[8] I. Garcia-Vargas and R. Senhadji-Navarro, "Finite state machines with input multiplexing: a performance study," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 5, pp. 867–871, 2015.

[9] L. Jozwiak, A. Slusarczyk, and D. Gawlowski, "Multi-objective optimal FSM state assignment," in *Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD'06)*, pp. 385–396, IEEE, Dubrovnik, Croatia, 2006.

[10] S. Deniziak and M. Wiśniewski, "FPGA-based state encoding using symbolic functional decomposition," *IEEE Electronics Letters*, vol. 46, no. 19, pp. 1316–1318, 2010.

[11] R. Senhadji-Navaro and I. Garcia-Vargas, "high-speed and area-efficient reconfigurable multiplexer bank for RAM-based finite state machine implementations," *Journal of Circuits, Systems and Computers*, vol. 24, no. 7, Article ID 1550101, pp. 1–15, 2015.

[12] M. Kołopieńczyk, L. Titarenko, and A. Barkalov, "Design of EMB-based moore FSMs," *Journal of Circuits, Systems and Computers*, vol. 26, no. 7, pp. 1–23, 2017.

[13] M. Kolopienczyk, A. Barkalov, and L. Titarenko, "Hardware reduction for RAM-based moore FSMs," in *Proceedings of the 7th International Conference on Human System Interactions, HSI 2014*, pp. 255–260, IEEE, Costa da Caparica, Portugal, June 2014.

[14] https://people.engr.ncsu.edu/brglez/CBL/benchmark/.

[15] S. Devadas, H.-K. Ma, A. R. Newton, and A. Sangiovanni-Vincentelli, "Mustang: state assignment of finite state machines targeting multilevel logic implementations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 12, pp. 1290–1300, 1988.

[16] K. Kabyl, A. Berrachedi, and É. Sopena, "A note on the cubical dimension of new classes of binary trees," *Czechoslovak Mathematical Journal*, vol. 65, no. 1, pp. 151–160, 2015.

[17] M. Liu and H.-M. Liu, "Vertex-fault-tolerant cycles embedding on enhanced hypercube networks," *Acta Mathematicae Applicatae Sinica*, vol. 32, no. 1, pp. 187–198, 2016.

[18] G. Pavai and T. V. Geetha, "New crossover operators using dominance and co-dominance principles for faster convergence of genetic algorithms," *Soft Computing*, pp. 1–26, 2018.

[19] S. Ganjefar and M. Tofighi, "Optimization of quantum-inspired neural network using memetic algorithm for function approximation and chaotic time series prediction," *Neurocomputing*, vol. 291, pp. 175–186, 2018.

[20] H. Huang, L. Lv, S. Ye, and Z. Hao, "Particle swarm optimization with convergence speed controller for large-scale numerical optimization," *Soft Computing*, pp. 1–17, 2018.

[21] R. Knobloch, J. Mlýnek, and R. Srb, "The classic differential evolution algorithm and its convergence properties," *Applications of Mathematics*, vol. 62, no. 2, pp. 197–208, 2017.

[22] F. E. Curtis, "A penalty-interior-point algorithm for nonlinear constrained optimization," *Mathematical Programming Computation*, vol. 4, no. 2, pp. 181–209, 2012.

[23] P. Armand and R. Omheni, "A mixed logarithmic barrier-augmented Lagrangian method for nonlinear optimization," *Journal of Optimization Theory and Applications*, vol. 173, no. 2, pp. 523–547, 2017.

[24] W. Murray and K.-M. Ng, "An algorithm for nonlinear optimization problems with binary variables," *Computational Optimization and Applications*, vol. 47, no. 2, pp. 257–288, 2010.

[25] E. M. Soler, V. A. De Sousa, and G. R. M. Da Costa, "A modified primal-dual logarithmic-barrier method for solving the optimal power flow problem with discrete and continuous control variables," *European Journal of Operational Research*, vol. 222, no. 3, pp. 616–622, 2012.

[26] R. Gárciga Otero and A. Iusem, "A proximal method with logarithmic barrier for nonlinear complementarity problems," *Journal of Global Optimization*, vol. 64, no. 4, pp. 663–678, 2016.

[27] L. Menniche and D. Benterki, "A logarithmic barrier approach for linear programming," *Journal of Computational and Applied Mathematics*, vol. 312, pp. 267–275, 2016.

[28] R. Shen, Z. Meng, C. Dang, and M. Jiang, "Algorithm of barrier objective penalty function," *Numerical Functional Analysis and Optimization*, vol. 38, no. 11, pp. 1–17, 2017.

[29] I. Chakroun, T. Haber, and T. J. Ashby, "SW-SGD: the sliding window stochastic gradient descent algorithm," *Procedia Computer Science*, vol. 108, pp. 2318–2322, 2017.

[30] A. Senov and O. Granichin, "Projective approximation based gradient descent modification," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3899–3904, 2017.

[31] B. Mu, J. Ren, and S. Yuan, "An efficient approach based on the gradient definition for solving conditional nonlinear optimal perturbation," *Mathematical Problems in Engineering*, vol. 2017, Article ID 3208431, 10 pages, 2017.