Journal of Engineering and Applied Sciences 11 (7): 1514-1524, 2016
ISSN: 1816-949X

Distributed Reconfiguration Algorithm for Self-Repairing in Cell-Based Architecture

Chanin Wongyai and Pradondet Nilagupta
Department of Computer Engineering, Kasetsart University, Bangkok, Thailand

Abstract: This study proposed the distributed reconfiguration algorithm that used for reconfiguring internal functions in cell-based hardware system when faults occur in a system. A traditional reconfigure mechanism is an external circuit to transfer new configuration bits into a system which a weak point in fault tolerant perspective. Consequently, we proposed an alternative reconfiguration mechanism that combines a functional unit and a distributed reconfiguration control unit. The control unit is operated by the distributed reconfiguration algorithm for recovering faults in internal functions. The control unit can detect a fault, diagnose status of adjacent control unit and functional modules and take control over on a faulty control unit and reconfigure functional modules for recovering faults in adjacent units. This paper shows how the proposed mechanism can use to recover a single fault and multiple faults. The experimental result shows 80-100% repair rates on simple circuits and ITC'99 benchmark circuits.

Key words: Fault-tolerence, evolve system, cell architecture, self-repair, fault recovery

INTRODUCTION

Reconfiguration system has capable of change functionality. An objective of reconfigured system is to change internal operations to support several situations. A fault in a system is one of the situations that the reconfiguration system has been designed to support that. The reconfiguration system that designed for supporting the fault in the system that called fault tolerant system.

In previous works, fault-tolerant models for reconfiguration system have been proposed, Virtual Reconfiguration Circuit (VRC) (Dhanasekaran *et al*., 2006; Kumar *et al*., 2007) is implemented similarly to traditional programmable logic circuits such as FPGA. They used Genetic Algorithms (GA) to develop a circuit that can handle faults in the VRC. The cell-based architectures are an evolving hardware that inspired by biological cell structure. Ferreiro and Rolle presented the neural network sarchitectures and a training algorithm that can predict properties associated with a performance of system. This model is a combination of system recovery strategy with artificial neural network technology for fault recovery. The cell-based architectures that were proposed in many works such as the embryonic project (Ortega and Tyrrell, 1999; Mange *et al*., 1998; Tempesti *et al*., 2007) are 2-dimentional array architecture with a row/column/cell elimination strategy for fault recovery. The honeycomb architecture (Tyrrell and Sun, 2006) is a 2-dimentional structure that each cell has six neighboring cells. The honeycomb architecture uses an evolutionary algorithm that based on chemical diffusion technique to manipulate information bits for fault recovery. An artificial cell (Szasz and Chindris, 2007) is four-level embryonic structure. An artificial cell uses a spare cell in a cluster to replace a faulty cell. The prokaryotic bio-inspired system (Samie *et al*., 2009) was proposed architecture that similar to biological prokaryotic cells. The prokaryotic system uses bus-based cell elimination strategy that information of a faulty cell will shift along the bus to spare cells to recovery faults. Typically, the cell architecture consists of a functional unit, a control unit and an interconnection unit. The functional unit is the main operation of a system that can change from one operation to another operation by reconfiguring from the control unit. The control unit is an intrinsic reconfiguration controller that used to change information of functional and interconnection module. The interconnection unit use to communicate with functional units and functional or control units. In previous works, the cell-based architectures were designed to support fault recovery only in the functional unit by assuming that the control and the interconnection units are fault-free.

In this study, we proposed the cell-based hardware architecture and distributed reconfiguration algorithms for a fault recovery. We focus on a fault that occurs in a control unit of cell-based architecture. Therefore, when the control unit failed, faults in the functional unit cannot

Corresponding Author: Chanin Wongyai, Department of Computer Engineering, Kasetsart University, Bangkok, Thailand

be recovered. To solve this problem, we consider the cell system into two parts: a functional unit and a control unit. A fault in each cell can be repaired and detected by its and other cells. The proposed control unit is multi-controls communication that each unit can be used to detect faults in its functional unit, a functional and a control unit of neighboring cells. For functional unit failure recovery, the control unit can perform self-repair algorithms to determine spare function unit's location and transfer information of the faulty function unit to the spare unit. For control unit failure recovery, we provided control algorithms that a control unit can attach to neighboring cells and transfer its information into those cells. When control unit is faulty, cells that received information will be active for controlling its function unit instead.

MATERIALS AND METHODS

Cell model: The system architecture is 2-dimensional as shown in Fig. 1 consists of M rows x N columns. Let i, j be a coordinate of a cell at row i and column j. Let C be a set of cell elements. Let A be a set of cells are occupied by an application circuit A⊂C. Let S be a set of spare cells that S⊂C and S≠A, |S| = |C|-|A|. Let CS be a set of the status ofcells, CS = {occupied, faulty, spare}.

Each cell consists of a control unit, a functional unit and shortest distance information which the values are a minimal distance from all spares cell to a cell on each direction. Let $cu_{i,j}$ be a control unit at row i and column j. Let $fu_{i,j}$ be a functional unit at row i and column j. Let $NC_{i,j}$ be a set of available neighboring control units of cell i,j, $Nc_i = \{cu_{i+1,j}, cu_{i-1,j}, cu_{i,j+1}, cu_{i,j-1}\}$. Let $n_{i,j}$be a shortest distance value from all spares cell on the north direction to cell i, j. Let $e_{i,j}$,$s_{i,j}$ and $w_{i,j}$ be a shortest distance value of east, south and west direction respectively. The shortest distance value in each direction of a control unit can determines by Eq. 1-4:

$$n_{i,j} = \min (n_{i,j}, \min(n_{i+1,j}, e_{i+1,j}, w_{i+1,j}) +1) \quad (1)$$

$$e_{i,j} = \min (e_{i,j}, \min(n_{i,j+1}, e_{i,j+1}, s_{i,j+1}) +1) \quad (2)$$

$$s_{i,j} = \min (s_{i,j}, \min(e_{i-1,j}, s_{i-1,j}, w_{i-1,j}) +1) \quad (3)$$

$$w_{i,j} = \min (w_{i,j}, \min(n_{i,j-1}, s_{i,j-1}, w_{i,j-1}) +1) \quad (4)$$

Fault problem: We classify faults in the system into 3 types: a functional unit fault thata fault occurs at a functional unit, a neighboring control unit fault that a fault occurs at neighboring control unit but a neighboring functional unit is fault-free anda neighboring cell fault that functional and control units of neighboring cell are faulty.

From fault types as mention above, the control unit fault is the important problem of the system. When a control unit detects a fault that occurs at neighboring control units, the control unit will perform operations that correspond to one of 4 cases of the candidate selection rule (rule 1) depend on positions of a faulty neighboring control unit. For a functional unit fault, the control unit of a faulty functional unit will find a spare cell for recovering a functional unit fault by sending finding signals that correspond to the finding rule (rule 2). In a case of multiple faults, that faults occur at 2 or more cells at the same time. Therefore, control units of faulty cells have to find spare cells by sending finding signals simultaneously. Any cells that received finding signals will decide to select one of finding signals which is propagated to neighboring cells and cancel other signals by using the propagation and cancelation rule (rule 3 and 4)

Candidate selection rule

Rule 1: This rule is used to select an appropriate control unit that will operate instead of a faulty control unit for the neighboring control unit fault. Let $Cf_{i,j}$ be a set of functional units which are controlled by a control unit $cu_{i,j}$. Aninitial element of $Cf_{i,j}=\{fu_{i,j}\}$. When neighboring control unit failed, a functional unit (fu) which its control unit is faulty will be added to $Cf_{i,j}$ by the rule as follows:

- Case 1: a control unit at the North direction is faulty:

$$CF_{i,j} \cup \{fu_{i+1,j}\} \text{ iff } s_{i+1,j} \le \min(n_{i+1,j}, e_{i+1,j}, w_{i+1,j}) \quad (5)$$

- Case 2: a control unit at the East direction is faulty:

$$CF_{i,j} \cup \{fu_{i,j+1}\} \text{ iff } w_{i,j+1} \le \min(n_{i,j+1}, e_{i,j+1}, s_{i,j+1}) \quad (6)$$

- Case 3: a control unit at the South direction is faulty:

$$CF_{i,j} \cup \{fu_{i-1,j}\} \text{ iff } n_{i-1,j} \le \min(e_{i-1,j}, s_{i-1,j}, w_{i-1,j}) \quad (7)$$

- Case 4: a control unit at the West direction is faulty:

$$CF_{i,j} \cup \{fu_{i,j-1}\} \text{ iff } e_{i,j-1} \le \min(n_{i,j-1}, s_{i,j-1}, w_{i,j-1}) \quad (8)$$

Finding rule: The finding rule is taken when a fault occurred in a functional unit or a cell. The rule 2 is a

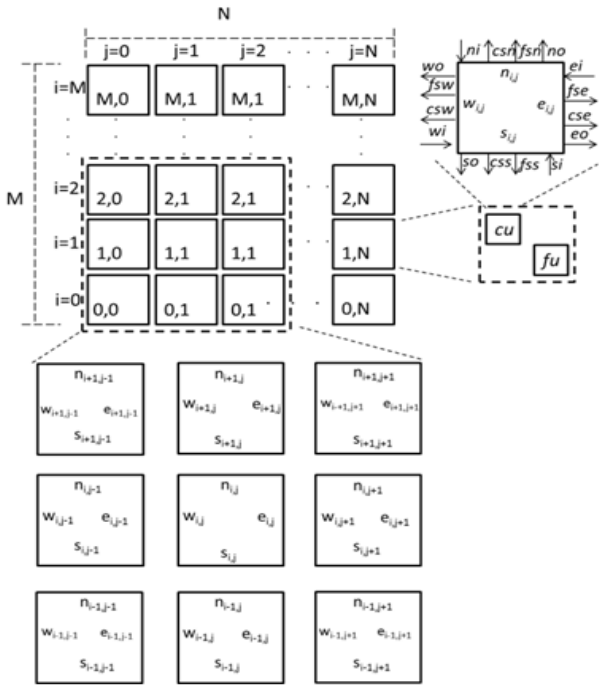

Fig. 1: The cell-based architecture model

procedure that generates finding signals via neighboring cells to find paths to spare cells. Equation 9-12 determine directions that send a finding signal. Let $fsn_{i,j}$ be a finding signal in the North direction. Let $fse_{i,j}$ be a finding signal in the East direction. Let $fss_{i,j}$ be a finding signal in the South direction. Let $fsw_{i,j}$ be a finding signal in the West direction.

Rule 2: Function unit fault of cell i, j is faulty:

$$fsn_{i,j} = \begin{Bmatrix} 1 \mid n_{i,j} \le \min(e_{i,j}, s_{i,j}, w_{i,j}) \\ 0 \mid \text{others} \end{Bmatrix} \quad (9)$$

$$fse_{i,j} = \begin{Bmatrix} 1 \mid e_{i,j} \le \min(n_{i,j}, s_{i,j}, w_{i,j}) \\ 0 \mid \text{others} \end{Bmatrix} \quad (10)$$

$$fss_{i,j} = \begin{Bmatrix} 1 \mid ds_{i,j} \le \min(n_{i,j}, e_{i,j}, w_{i,j}) \\ 0 \mid \text{others} \end{Bmatrix} \quad (11)$$

$$fsw_{i,j} = \begin{Bmatrix} 1 \mid w_{i,j} \le \min(n_{i,j}, e_{i,j}, s_{i,j}) \\ 0 \mid \text{others} \end{Bmatrix} \quad (12)$$

Propagation and cancelation rule: This is a decision rule to select one of finding signals which is propagated to neighboring cells and cancels other signals when 2 or more signals has arrived the cell at the same time. Let $csn_{i,j}$ be a canceled signal in the North direction. Let $cse_{i,j}$ be a canceled signal in the east direction. Let css i, j be a canceled signal in the south direction. Let $csw_{i,j}$ be a canceled signal in the west direction. Let $Wd_{i,j}$ be a weight of shortest distance value in all direction of a cell i, j. Let R be the number of distance values which equal to 0. A control unit will operate by using the rule of propagationand cancelation as follows:

Rule 3: The number of an incoming finding signal = 1. A control unit uses Eq. 9-12 for propagating finding signals.

Rule 4: The number of incoming finding signals = 2. A control unit uses Eq. 9-12 for propagating finding signals. For canceling signal, a direction which is canceled can determine byequation Eq. 13-17:

$$Wd_{i,j} = \frac{n_{i,j} + e_{i,j} + s_{i,j} + w_{i,j}}{R+1} \quad (13)$$

$$csn_{i,j} = \begin{Bmatrix} 1 \mid Wd_{i+1,j} > \min(Wd_{i,j+1}, Wd_{i-1,j}, Wd_{i,j-1}) \\ 0 \mid \text{others} \end{Bmatrix} \quad (14)$$

$$cse_{i,j} = \begin{Bmatrix} 1 \mid Wd_{i+1,j} > \min(Wd_{i+1,j}, Wd_{i-1,j}, Wd_{i,j-1}) \\ 0 \mid \text{others} \end{Bmatrix} \quad (15)$$

$$css_{i,j} = \begin{Bmatrix} 1 \mid Wd_{i-1,j} > \min(Wd_{i+1,j}, Wd_{i,j+1}, Wd_{i,j-1}) \\ 0 \mid others \end{Bmatrix} \quad (16)$$

$$csw_{i,j} = \begin{Bmatrix} 1 \mid Wd_{i,j-1} > \min(Wd_{i+1,j}, Wd_{i,j+1}, Wd_{i-1,j}) \\ 0 \mid others \end{Bmatrix} \quad (17)$$

An algorithm: From cell architecture, the proposed structure has 2 main parts: a functional unit and a control unit. The functionalunitwill perform normal operations. The controlunit will use to recon figure a functional unit and diagnose status of all neighboring cells. Each control unit will execute the control algorithm that is decomposed into 3 phases as follow:

Phase 1: nitialize phase
Step 1: A cell that is a spare generates an initialed distance value to all neighboring cells
Step 2: A cell received distance values from all neighboring cells.
Step 3: A cellcalculates and stores a shortest distance value of each direction (n,e,s,w) and propagatesan incremental distance valueto all neighboring cells
Phase 2: operation phase
Step 1: A control unit configures its functional unit with initial configuration bits
Step 2: Recalculate configuration bits of interconnection with a coordinate of a neighboring cell and transfers to neighboring cells
Step 3: A control unit holds in the normal state for detecting control signals and fault signals
Phase 3: fault phase
Step 1: Detecting and diagnose fault signals from all neighboring cells
Step 2: Classifying a fault type
Control fault
Case 1: Neighboring control units = 2 ($|NC_i| = 2$)
The functional unit of the faulty control cell has taken control over by a control unit with a lowest distance value corresponds to the candidate selection rule R1
Case 2: Neighboring control units <2 ($|NC_i| < 2$)
The functional unit is marked as faulty
A control unit with a lowest distance value takes controlthefaulty functional unit corresponds to the rule 1
The new control finds paths to spare cells by sending finding signal to neighbor cell with the rule 2
Each control that received finding signals will propagate the signals to neighbor cell with the rule 2
Transfer configuration bits to a new cell on the path of a spare cell.
Recalculate configuration bits of interconnection
Functional fault:
The control finds paths to spare cellsby sending finding signal to neighbor cell with the rule 2
Each control that received finding signals will propagate the signals to neighbor cell with the rule 3 or 4
Transfer configuration bits to a new cell on the path of a spare cell
Recalculate configuration bits of interconnection
If repairing complete, then generate complete signals to all cells and repeat phase 1

The controlalgorithm will be distributed to all cells. Each control unit will use to send/receive fault signals to/from other cells and perform the control algorithms for finding and replacing the faulty units. From the algorithm, there consists of 3 phases. In the first phase, cells that are a spare cell will generate distance value to neighboring cells. Each cell will determine and store shortest distance values as shown in Fig. 2.

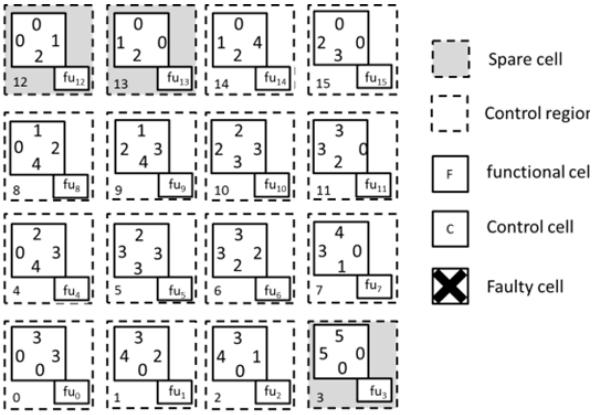

Fig. 2: Initializing distance values to all cells

In the second phase, the control unit will take control a functional cell and transfers configuration bits to a functional cell. After that, the control unit will recalculate configuration bits of interconnection with a coordinate of neighboring cells and transfers configuration bits of interconnection and functional to all neighboring cells. In the third phase, an operation of a control unit will operate depends on the fault types: the control fault and functional fault. When all operation has done, phase 1 is repeated for updating shortest distance value.We will give examples of the self-repair strategy for the 3 fault scenarios.

The functional unit fault: When a control unit detects a fault in its functional unit, thecontrol unit will generate finding signals and providethat signals to neighboring cells with lowest distance value to locate spare functional units. When the control unit of neighboring cell received finding signals, the control unit will propagate the finding signals to the neighboring cells correspond to the rule 3 or 4. If a finding signal reaches a spare cell, the control unit will provide a backtracking signal to the previous cells in propagation paths. When the control unit receives the backtracking signal, the control unit will reconfigure its functional unit with information of the neighboring cell that sent the finding signal as shown in Fig. 3.

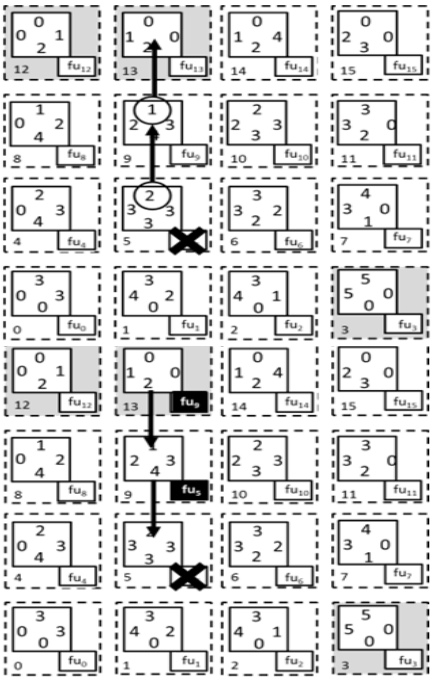

Fig. 3: A Functional unit fault recovery

The control unit fault: This fault appears in the system when a fault occurs in the control unit (functional unit is fault-free). The neighboring control units can identify a fault signal that informed from an error checker of a faulty control unit.All neighboring control unit will wait for the takeover signals from the functional unit of the faulty cell. In the functional unit, the number of available neighboring control units is determined by the control selection. If the cell has available neighboring control units <2 units, the control selection will send a takeover signal to the selected neighboring cell. The candidate selection rule is used to choose the appropriate control unit. When a control unit has received a takeover signal, the control unit will take a functional unit into a control cluster (Lakamraju and Tessier, 2000) as shown in Fig. 4 called the control region. For the cell has available neighboring control units <2 units, the control selection will set its functional unit as faulty cell and send a fault signal to available neighboring control units. A control unit of a cell that has received the fault signal will move from the wait state to finding state and propagate the finding state to neighboring cells to locate spare cells. This finding technique, we apply the maze finding method (Lee, 1961). The cell that is in the finding path will reconfigure the configuration memory with configuration bits of neighboring cell. Figure 5 shows the self-repairing methodology for control unit fault when the cell has available neighboring control units l<2 units.

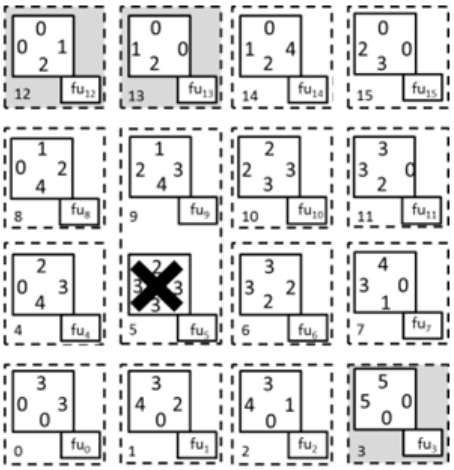

Fig. 4: Local controlling

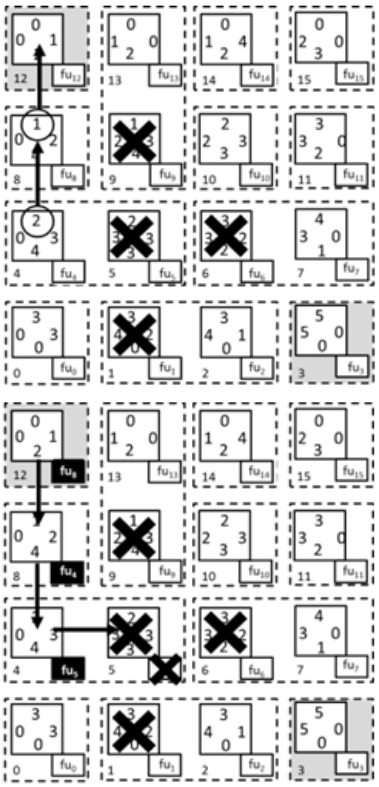

Fig. 5: A control unit repairing when neighboring control units <2 units

The functional unit and neighboring unit fault: This fault is multiple fault types that faults will occur in its functional unit and a neighboring control unit at the same time. Therefore, the control unitwill operate similarly to bold the functional unitfault and the neighboring control unit fault as shown in Fig. 6. First, the control unit will provide the finding signal to locate a spare cell and

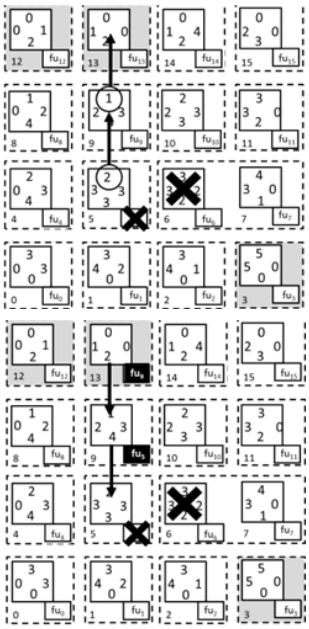

Fig. 6: a Functional unit and neighboring control unit fault recovery

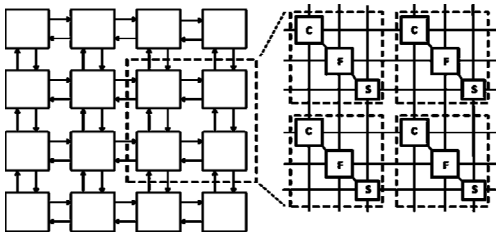

Fig. 7: Cell-based hardware architecture

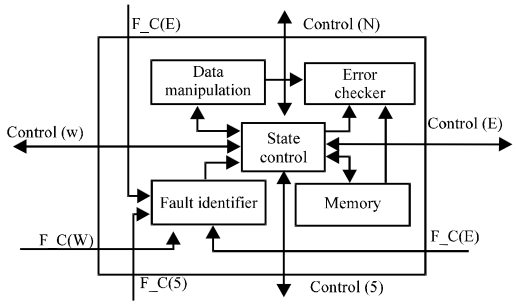

Fig. 8: The control unit architecture

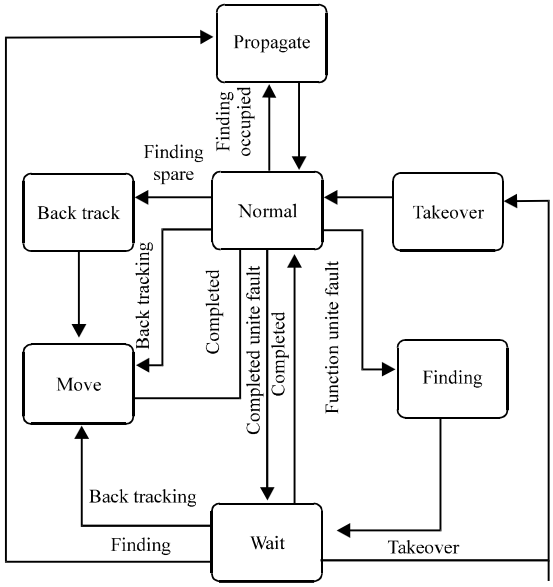

Fig. 9: Control state machine

reconfigure cells along the finding path. Second, the control unit that has the lowest distance from the spare cell will take over a faulty control unit instead.

Hardware architecture: The cell-based hardware architecture is inspired from a structure of the biological cell. Organization of cell-based is a 2-dimensional architecture that consists of identical cells. A cell structure consists of 3 parts: Control units (C), Functional units (F) and interconnection units (S) as shown in Fig. 7.

The control unit: The control unit uses to control and reconfigure functional unit and interconnection unit. The control unit will perform a control algorithm to detect and diagnose faults. In addition, the control unit will identify a status of the functional unit and neighboring control units and uses to reconfigure functional units and interconnection when faults occurred. The control unit will executethe reconfiguration algorithm and diagnose a status and information (e.g., shortest distance values, finding signals and canceled signals) of neighboring control and functional units via control signals (Control (N). (E), (S), (W)) and fault signals (F_C (N), (E), (S), (W)). Each control cell consists of five parts: State Machine control (SM), Fault Identifier (FI), Error Checker (EC), control memory and data manipulation moduleas shown in Fig. 8. The state machine control is used to control the reconfiguration algorithm. By finding a spare cell and reconfiguring of the functional unit of the neighboring cells and its functional unit. The state machine consists of 7 states that operate according to the type of faults, control signals and the cell status as shown in Fig. 9.

Normal state: There is a start state that performs an initial operation and identifies a type of fault signal or control signals. The state machine will operate according to the type of faults and control signals. If a functional unit fault or neighboring functional unit fault occurs, the state will

move to the finding state that generates finding signals to neighboring cells. If the fault type isa control unit fault, the state will move to the wait state. For a normal operation, the control will wait for the finding signals. If a control has received the finding signal, the state will identify the cell status. If the cell status is spare, the state will move to the backtrack state. If the cell status is occupying, the state willmove to the propagate state.

Finding state: There is a state that generates the finding signals to neighboring cells in order to locate spare cells.

Wait state: There is a state that identifiesa control signal. If the control signal is a backtracking signal, the state willmove to the move state that the cell is activated with the configuration of a neighboring cell or faulty cell. If the control signal is a takeover signal, the state is moved to the takeover state.

Takeover state: There is a state thatset a control active bit in configuration memory for taking control a faulty functional unit instead.

Backtracking state: There is a state that sends the backtracking signal back to source cells and moves to the move state.

Move state: There is a state that uses to reconfigure the operations of the cell. The cell will be activated with the configuration of a neighboring cell which was sent the finding signal.

Propagate state: There is a state that passes through the finding signal to the neighboring cells if the cell status is occupying. In this way, the signal will be propagated to all neighbor cells except which was sent the finding signal. For finding signals have arrived 2 or more, a control unit will operate by using the rule of propagating. The error checker is responsible to diagnose sequence of states of control units and a fault from its and neighboring functional units. By this method, we provide two parts of fault checkers: state sequence checker and combinational checker. The error checker is able to detect faults in the state machine control, the configuration memory and data manipulation. For state machine control, the state checker has implemented as state outputs and next state checker. The state line outputs will be encoded byberger code and compared by the 2-rail logic of the circuits (Mokhov *et al*., 2012). The 2-rail logic is used to compare the inputs and output of the logic element (codeword) with the complement inputs and the outputs from duplicated logic in the control unit. The configuration memory

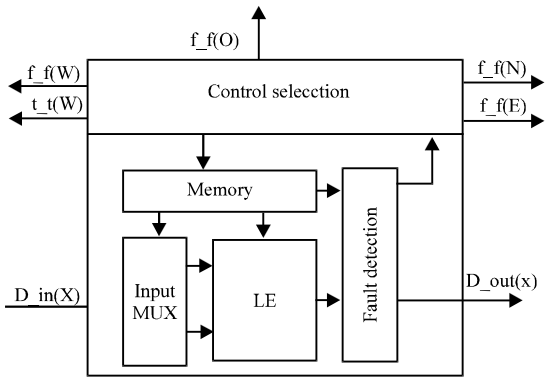

Fig. 10: The functional unit architecture

checker has implemented as parity checker. Data manipulation checker has implemented as combinational checker that based on the 2-rail logic scheme. The checker will generate fault signals and inform tothe state machine control.

The fault identifier has designed as fault queuing of surrounding cells that ordering fault signals from neighboring cells and encodes fault signals into a fault type. The fault type signal is identified as functional unit fault or control unit fault after that will inform to a state machine control.

The control memory is used to store configuration bits of the functional unit. In each cell, not only to store configuration bits of its functional unit but also store configuration bits of the neighboring functional units in a North, East, South and westposition. The configuration bit of the neighboring units is used as the backup data if the neighboring cells are faulty. Each word of the configuration bit is similar to the configuration bits in the functional units and local shortest distance information.

The data manipulation will use to modify configuration bits of cells that moved to a new cell. This module will modify the configuration bits of interconnection that corresponding to a current position and new position. The configuration of interconnection is used to control interconnection multiplexers. In addition, the data manipulation is also used to change the functionality of cell from logic function to a router (i.e., buffer). The router is used to extend signal paths when the distance between the source cell and the target cell longer than the interconnectionof a cell.

The functional unit: The functional unit is the main functionality of the cell system that performs operations of the system. The functional unit consists of five parts: the logic element, the input multiplexer, the configuration memory, the control selection and the fault detectionas shown in Fig. 10. The logic element is designed as 2-inputs Look-Up Table (LUT) with a flip-flop that can

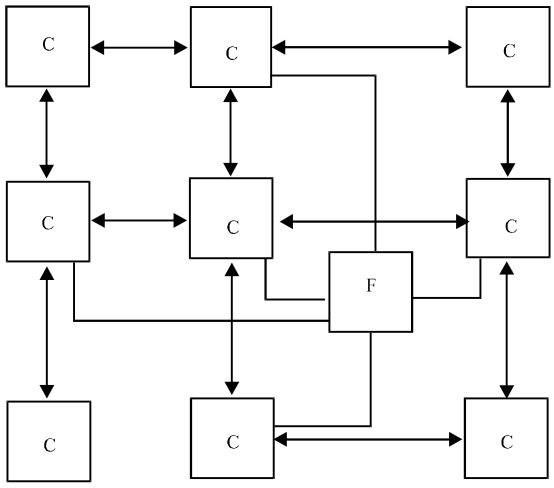

Fig. 11: Controls and a functional unit communication

perform a simple logic function such as AND, OR, EX-OR, NOT or sequential circuits. Inputs of a logic element are connected to the input multiplexer that select data from North, East, South and west cell, a feedback input from its flip-flop and routing switches. Outputs of the logic element are connected to all neighboring cells (i.e., North, East, South and West) and routing switches that use to send/receive to non-neighboring cells.

The configuration memory is a 16 bit memory that consists of 2-5 bits of an input multiplexer, 4 bit of a logic element and 2 bit of parity. The configuration memory is used to store configuration bits of the functional unit. The control selection is used to detect a fault signal from its control units and inform a takeover signal to neighboring control units. When its control unit failed, all functional unit signals (f_f) switched to a neighboring control cell.

The interconnection unit: For an interconnections structure,we organize a structure of cell as 2 parts: functional unit and control unit. Functional units are organized as a functional layer. Control units are organized as a control layer. In each layer, the units can communicate with each other and also communicate with the units in the different layer. As shown in Fig. 11, the neighboring control units can communicate with functional unit for reconfiguring and detecting a status of functional unit. Communication between functional units is to establish the connection to make an application circuit. Communication between control units is to make a network of fault detection and reconfiguration control as shown in Fig. 12.

Table 1: Benchmark circuits configuration

Circuits	Work cells	Spare cells
Counter	18	46
Multiply	42	22
b01	45	19
b02	24	40
b06	51	13

RESULTS AND DISCUSSION

We use Xilinx ISE tool to synthesize the model that consists of cell size of 8×8. We use a simple 4 bits binary counter circuit, 4 bits multiply and 3 circuits of ITC'99 benchmark in our experimental. Table 1 shows the number of cells that used for each circuit and the number of spare cells. In the experiment, we classify fault scenarios into 3 types: a fault occurs in the Function Unit without Fault in the control unit (FF), a fault occurs in the control unit without fault in the Functions Unit (CF), faults occur in the function unit and the Control Unit (FCF) we called cell fault. To evaluate repair rate of benchmark circuits, we injected fault (s) at control units and also functional units. In addition, we injected fault(s) of 1 to the number of spares by randomizing locations.

Figure 13-15 show the repair rate of benchmark circuits in the different number of faults with randomly fault positions. Each point is an average of 100 test cases with different fault pattern. Repair rate of each test case was calculated by Eq. 18:

$$\text{Repair rate} = \frac{\text{Number of repairable faults}}{\text{Number of faults}} \times 100 \qquad (18)$$

Figure 13a-e show the repair rate of benchmark circuits for functional unit fault Fig. 14 and 15 show the repair rate for control unit fault and both functional and control unit fault respectively. In functional unit fault, we compare the proposed method with cell eliminated. To evaluate repair rate we injected fault in 2 types: Single (SF) and Multiple Faults (MF). Spare cells location is randomly distributed for the proposed method and is row/column at topmost row/rightmost column for cell eliminated method. We provided a 1 spare row and 1 spare column for b01 and bo6circuits, 3 spare rows and 3 spare columns for counter and b02 circuits and 1 spare row and 2 spare columns for the multiply circuit.

In functional unit fault, the proposed model can reach the repair rate nearly 100% when we inject faults is equal to a number of spares in single fault type. In multiple faults, the repair rate decreases to 80% when injected fault is equal to the number of spares. Since, the complexity of connections in each circuit, some spare cells use to

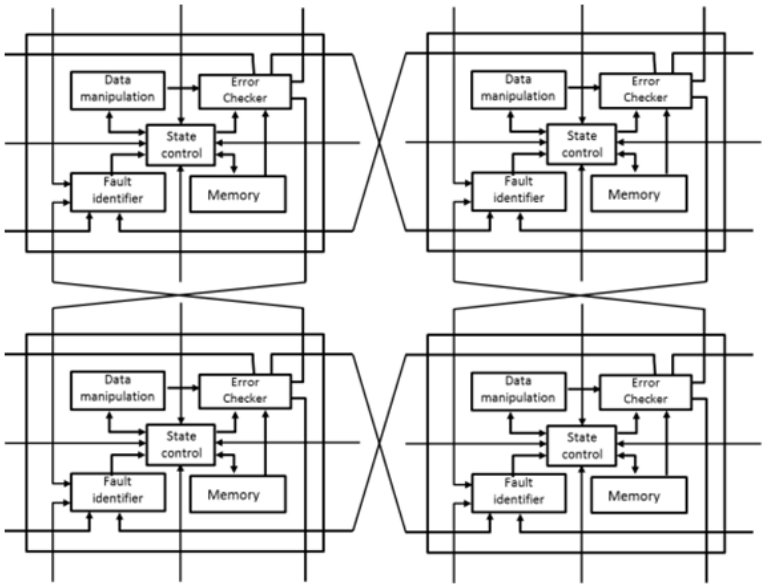

Fig. 12: Communicating between control units

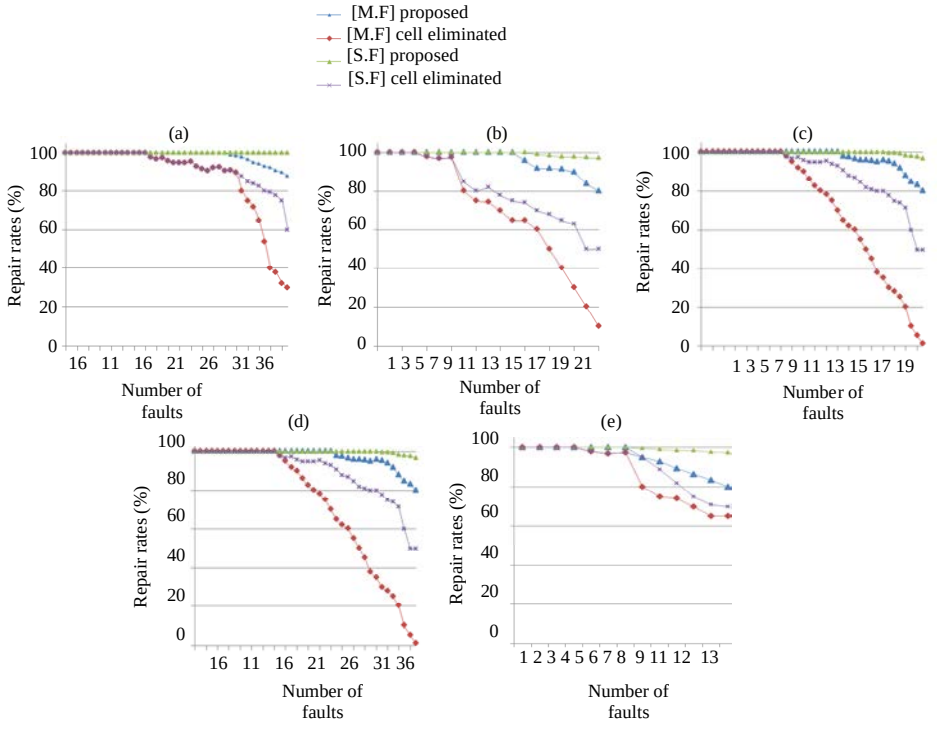

Fig. 13: Functional unit fault; a) counter; b) multiply; c) b01; d) b02 and e) b06

support routing connection,therefore, spare cells cannot use to recover faults as equal as the number of spares. In control unit fault, we did not compare with cell eliminate since that method is not able to recover faults in control units. The proposed model can keep the repair rateat 100% until 16 faults and then slowly decreasingbecause some fault pattern cannot be repaired. In a b06 benchmark, the circuit has only spare cells of 13 but keeps 100% of the repair rate to 16 faults because the candidate control scheme improves controllability and repairability of

Table 2: Utilization of spare cells (Single fault)

	FF					
Circuits	No. of faults	Proposed (%)	No. faults	CE (%)	CF proposed (%)	FCF proposed (%)
Counter	46	100.0	15	32.6	34.7	34.7
Multiply	16	77.2	5	22.7	72.7	72.7
b01	13	68.4	4	21.0	84.2	68.4
b02	33	82.5	15	37.5	40.0	40.0
b06	7	53.8	4	30.7	100.0	53.8

Table 3: Utilization of spare cells (Multiple faults)

	FF					
Circuits	No. of faults	Proposed (%)	No. faults	CE (%)	CF proposed (%)	FCF proposed (%)
Counter	29	72.5	15	32.6	34.7	34.7
Multiply	16	72.7	5	22.7	72.7	72.7
b01	12	63.1	4	21.0	84.2	68.4
b02	25	62.5	15	37.5	40.0	40.0
b06	7	53.8	4	30.7	100.0	53.8

Table 4: Utilization of sparecells

	CF		FCF	
Circuits	No. faults	Proposed (%)	No. faults	Proposed (%)
Counter	15	34.7	15	34.7
Multiply	15	72.7	15	72.7
b01	15	84.2	13	68.4
b02	15	40.0	13	40.0
b06	15	100.0	7	100.0

system. The proposed model can use adjacency control units to recover a fault instead of a spare cell. Therefore, this model can repair control units more than the number of the spare cell.

In functional and control unit fault, the trend of repair rate is not proportional to the number of faults. But the proposed model can keep the repair rate in the range 80-100%. After analyzing this result, we found that fault patterns have an effect on the repair rate. In cell-based architecture have interconnections between cells by using the direct connection and non-direct connection. Any fault patterns which can completely cut the circuit into 2 parts,as a result, circuits cannot communicate, these faults cannot be repaired.

For the fault-tolerant system, 100% repair rate is a satisfied condition for repairing system. To evaluate an efficiency of the model, we measured the number of repairable fault at 100% repair rate and compared with the number of spares. Therefore, we provided an indicator to measure an efficiency of the model, calledthe utilization of spare. Table 2-4 show the utilization of spare cells at 100% repair rate is that an average of the utilization of each circuit. The utilization of spare cells was calculated by Eq. 19

$$\text{Utilization} = \frac{\text{Number of repairable faults}}{\text{Number of spares}} \times 100 \quad (19)$$

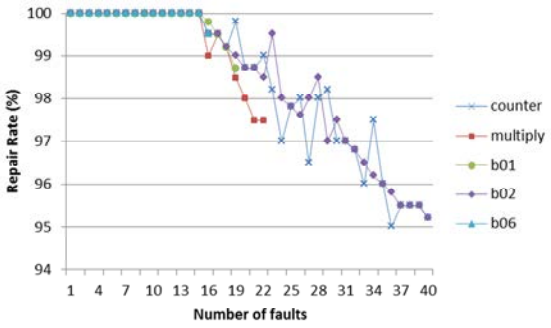

Fig. 14: Control unit fault

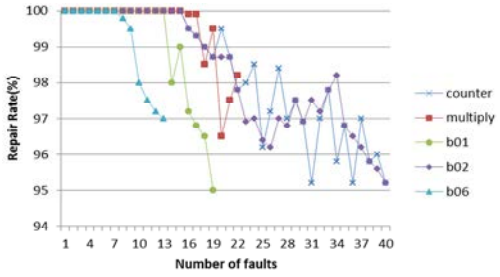

Fig. 15: Functional and control unit fault

From the result, in single fault, the utilization of spare of the proposed model is 100, 77.2, 68.4, 82.5 and 53.5% for circuit counter, multiply, b01, b02 and b06, respectively. For functional unit fault, the proposed model utilizes spares effectively than the cell eliminated model. For control unit fault, the proposed model can reach 100% the utilization with few spare cells such that the circuit b06 can recover faults of 16 with spare cells of 13. In functional and control unit fault, the model give the utilization of spare cells is less than the control unit fault but can keep an average ofthe utilization >50%.

CONCLUSION

The cell architectureshave been proposed that can tolerant only in functional unit fault. The proposed architecture designed to support a control unit fault. From the cell architecture and the distributed reconfiguration algorithm that allows cells are able to detect faults and reconfigure all neighboring cells, the cells that divided into two parts are able to detect and repair the faults at control and functional units. Control units can use to reconfigure a functional unit to recover functionality of the system when a function unit is faulty. It can be used to take control over an adjacent functional unit that has faulty control unit. As a result, the cell that has a fault only in the control unit still operates with the neighbor control unit. Therefore, this fault recovery technique of the cell can improve the capable of repairing in control units in the cell system instead of a functional unit only. From experimental result, this technique can reach 80-100% of the repair rate on simple and complex benchmark circuits

REFERENCES

Dhanasekaran, D., K.B. Bagan and E. Ravi, 2006. Fault tolerant system design using evolved virtual reconfigurable circuit. Int. J. Comput. Sci. Network Secur. (IJCSNS.), 6: 64-72.

Kumar, P.N., S. Anandhi and J.R.P. Perinbam, 2007. Evolving virtual reconfigurable circuit for a fault tolerant system. Proceedings of the 2007 IEEE Congress on Evolutionary Computation, September 25-28, 2007, IEEE, Chennai, India, ISBN:978-1-4244-1339-3, pp: 1555-1561.

Lakamraju, V. and R. Tessier, 2000. Tolerating operational faults in cluster-based FPGAs. Proceedings of the 2000 ACM-SIGDA Eighth International Symposium on Field Programmable Gate Arrays, February 1-11, 2000, ACM, Monterey, California, ISBN:1-58113-193-3, pp: 187-194.

Lee, C.Y., 1961. An algorithm for path connections and its applications. Electron. Comput., IRE Trans., 3: 346-365.

Mange, D., E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal and C. Piguet, 1998. Embryonics: A new methodology for designing field-programmable gate arrays with self-repair and self-replicating properties. IEEE Trans. Very Large Scale Integr. Syst., 6: 387-399.

Mokhov, A., V. Khomenko, D. Sokolov and A. Yakovlev, 2012. On dual-rail control logic for enhanced circuit robustness. Proceedings of the 12th International Conference on Application of Concurrency to System Design (ACSD), June 27-29, 2012, IEEE, Newcastle Upon Tyne, England, ISBN:978-1-4673-1687-3, pp: 112-121.

Ortega, C. and A. Tyrrell, 1999. Reliability analysis in self-repairing embryonic systems. Proceedings of the First NASA-DoD Workshop on Evolvable Hardware, July 21, 1999, IEEE, Toronto, Ontario, ISBN:0-7695-0256-3, pp: 120-128.

Samie, M., G. Dragffy, A. Popescu, T. Pipe and J. Kiely, 2009. Prokaryotic bio-inspired system. Proceedings of the NASA-ESA Conference on Adaptive Hardware and Systems (AHS09), July 29-August 1, 2009, IEEE, Bristol, England, ISBN:978-0-7695-3714-6, pp: 171-178.

Szasz, C. and V. Chindris, 2007. Development strategy and implementation of a generalized model for FPGA-based artificial cell in bio-inspired hardware systems. Proceedings of the 2007 5th IEEE International Conference on Industrial Informatics, Vol. 2, June 23-27, 2007, IEEE, Cluj-Napoca, Romania, ISBN:978-1-4244-0850-4, pp: 639-643.

Tempesti, G., D. Mange, P.A. Mudry, J. Rossier and A. Stauffer, 2007. Self-replicating hardware for reliability: The embryonics project. ACM. J. Emerging Technol. Comput. Syst. (JETC.), Vol. 3, 10.1145/1265949.1265955.

Tyrrell, A.M. and H. Sun, 2006. A honeycomb development architecture for robust fault-tolerant design. Proceedings of the First NASA-ESA Conference on Adaptive Hardware and Systems (AHS'06), June 15-18, 2006, IEEE, Toronto, Ontario, Canada, ISBN:0-7695-2614-4, pp: 281-287.