Hindawi Publishing Corporation
Journal of Computational Engineering
Volume 2014, Article ID 793635, 13 pages
http://dx.doi.org/10.1155/2014/793635

Research Article

Analysis of Parallel Multidimensional Wave Digital Filtering Network on IBM Cell Broadband Engine

Chien Hsun Tseng

Department of Information Engineering, Kun Shan University, No. 195 Kun-Da Road, Yung-Kang, Tainan 71003, Taiwan

Correspondence should be addressed to Chien Hsun Tseng; jason.tseng.taiwan@gmail.com

Received 5 November 2013; Accepted 3 January 2014; Published 17 February 2014

Academic Editor: Delfim Soares Jr.

As an alternative approach for the numerical integration of physical systems, the MDWDF technique has become of importance in the field of numerical analysis due to its attractive features, for example, massive parallelism and high accuracy both inherent in nature. In this study, speed-up efficiencies of a MDWDF network are studied for the linearized shallow water system, which plays an important role in fluid dynamics. To achieve the goal, the full parallelism of the MDWDF network is established in the first place based on the chained MD retiming technique. Following the implementation on the IBM Cell Broadband Engine (Cell/BE), excellent performance of the full parallel architecture is revealed. The IBM Cell/BE containing 1 power processor element (PPE) and 8 synergistic processor elements (SPEs) perfectly fits the architecture of the retimed MDWDF model. Empirical results have demonstrated that the full parallelized model with 8 processors (1PPE + 7SPEs) outperforms the other three models: partial right/left-loop retimed models and the full sequential model with 4× improvements for scheduled grids 51 × 51. In addition, for scheduled fine grids 201 × 201, the full parallel model is shown to possess significant performance over these models by up to 7× improvements.

1. Introduction

Physical system modeling is an important discipline in all fields as it enables the development of system simulation engines for physically realistic processes such as fluid flow, electrical, and acoustical phenomena. The most popular kind of models for multidimensional (MD) physical systems can be represented by sets of linear and/or nonlinear partial differential equations (PDEs) with properly imposed initial and boundary conditions. Many numerical approaches existing to analyze the behavior of such physical systems include finite elements (FEs) and finite differences (FDs). The FEs permit easy inclusion of local grid refinement and handling of complex geometries [1]. These methods, however, are computationally expensive and harder to directly and correctly set up the simulation plane than the frequently used finite difference methods. The latter approaches, on the other hand, have difficulties in handling irregular boundaries and need extra care of the local grid refinement in order to increase its measurement accuracy [2]. As FEs and FDs require a large number of grid storage in order to get results even at or around a point of interest with acceptable accuracy, the requirements for excessive CPU runtime and storage consumption are often unnecessarily large in such cases. Furthermore, the computational load of the difference method due to local grid refinement also prevents its use for real-time hardware synthesis.

Built on properties of the traveling wave formulation of lumped electrical elements to the modeling and simulation of a system represented by PDEs, a novel approach named wave digital filtering (WDF) network [3] had been proposed in the past due to its excellent features that fit requirements of practical interest [4, 5]. Unlike most types of digital filter, every delay element in a WDF network can be interpreted physically as holding the current state of a mass or spring (or capacitor or inductors) [3]. Furthermore, because the rules for interconnecting the elementary models are based on scattering theory, all signals explicitly computed via a WDF

network are interpreted as traveling wave components of physical variables [4, 6]. Making use of the WDF network paradigm and analogies with electrical networks, the MDWDF technique, thus, draws a maximum advantage of essential physical properties of such systems, in particular of causality, passivity, stability, finite propagation velocity, and so forth, which can be all translated to the actually considered physical problems so as to preserve important relationships between variables [4, 5, 7–10]. Proceeding in this way, it has the unique advantage of simultaneously offering the second-order accuracy, high robustness and fault tolerance, massive parallelism, full localness (also for taking into account arbitrary boundary conditions and shapes), and explicit or at least semiexplicit computability.

Fettweis and fellow researchers [3–5, 7] have carried out the pioneering work in the area of this subject. Our interest in the past and currently has revolved around the software toolbox development and hardware designing aspects [8–13]. In particular, the full parallel architecture of MDWDF network based on the chained MD retiming technique [14], a class of software pipelining, is focused on partitioned nested loop across distinct iterations in terms of spatial and temporal updating of its corresponding nonparallel MDWDF network. With the full parallelism adopted within loops, it is then desirable to exploit the multiple core hardware architecture with safe and efficient multithreading paradigm per core to further boost the performance of the MDWDF network. This will be a significant advantage of parallel implementation. Clearly, the proposed method is only attractive if it can be shown that the resultant simulation models provide an efficient technique to the solution of certain PDEs representing mechanical behaviors of the physical system when it is compared with conventional approaches such as finite elements.

Adopting the full parallel architecture [11], in this study, the analysis of CPU runtime speed-up efficiencies is empirically implemented on the IBM Cell Broadband Engine (Cell/BE) to further improve performance of the MDWDF network representing the linearized shallow water (LSW) system, which plays an important role in fluid dynamics. The IBM Cell/BE facilitates 1 power processor element (PPE) core processor providing system functions together with 8 synergistic processor element (SPE) coprocessors optimized for efficient data processing. As the key design goal of the full parallel MDWDF network is to maximize the performance in terms of speed-up efficiency, the IBM Cell/BE perfectly fits the architecture of the retimed MDWDF network, which requires at most 1 main processor and 7 routine processors running independently. Empirical results have demonstrated that the full parallelized model making use of 8 processors (1PPE + 7SPEs) significantly outperforms other three models: models with partial right/left-loop retimed bodies and the full sequential model by at least 3× and 4× improvements for scheduled grids 51 × 51, respectively. In addition, for scheduled grids 201 × 201, the full parallel model is shown to possess significantly performance over these models by up to 7× improvements.

2. Summary of 2D LSW System and Modeling Techniques

2.1. 2D LSW System and Its Corresponding Multidimensional Wave Digital Filtering Network. Let us consider a linearized shallow water (LSW) system characterized by a set of PDEs for the surface displacement, η [1, 8, 16]:

$$\begin{aligned} \frac{\partial v_1}{\partial t} - f v_2 + g\frac{\partial \eta}{\partial x} &= 0, \\ \frac{\partial v_2}{\partial t} + f v_1 + g\frac{\partial \eta}{\partial y} &= 0, \\ H\frac{\partial v_1}{\partial x} + H\frac{\partial v_2}{\partial y} + \frac{\partial \eta}{\partial t} &= 0. \end{aligned} \tag{1}$$

Here the horizontal velocities v_1 and v_2 are directed along the spatial domain of x and y direction, respectively. Furthermore, h is the total water depth defined as the sum of the undisturbed water depth (a constant mean depth) H and the free surface elevation η measured upward from the undisturbed surface; that is, $h = H + \eta$. The gravity acceleration g and the Coriolis parameter f are constants. This hydrostatic formulation is the combined physical processes of radiation, refraction, diffraction, and reflection when the system is confined in a bounded domain with nonempty piecewise boundary. In an unbounded domain, the LSW system, however, generates divergent and nonreflective waves based on some artificial open boundaries [2]. As a result, this modeling is quite successful in fluid dynamics to predict the dynamics of the surface gravity waves and is usually applied to the global modeling on large scale oceanographic or atmospheric quantities like transports and surface elevation, Tsunami modeling, and simulation [2, 6].

Applying the standard procedure known from the MD wave digital filtering [4, 5] for transforming the set of PDEs to its equivalent discrete passive model, a lumped MD-passive Kirchhoff circuit (MDKC) and its discrete approximation of MDWDF network [11] depicted in Figures 1(a) and 1(b), respectively, are obtained for the numerical integration of the LSW system. Since the resulting network behaves in the same way as the continuous one, it also preserves passivity for the discrete dynamical system, thus ensuring full robustness and stability of the algorithm [4, 12]. The reader is referred to [8] for more details of converting the given physical system to form the MDWDF network via the MDKC.

2.2. Chained Multidimensional Retiming. The objective of the chained MD retiming technique applied to the MDWDF network of Figure 1(b) is to legally change the delay of edges in the MD data flow graph (MDFG) such that nonzero delays on all edges can be obtained in order to achieve the full parallelism of the MDWDF network. Here the chained MD retiming is based on the push up scheduling technique [14, 16] where a good example illustrated in Figure 2(b) represents a retimed MDFG from a valid MDFG depicted in Figure 2(a). We note that the form of MDFG is described as a cyclic data flow graph with the tuple (V, E, D, t) to represent a node-weighted and edge-weighted graph where V is the set

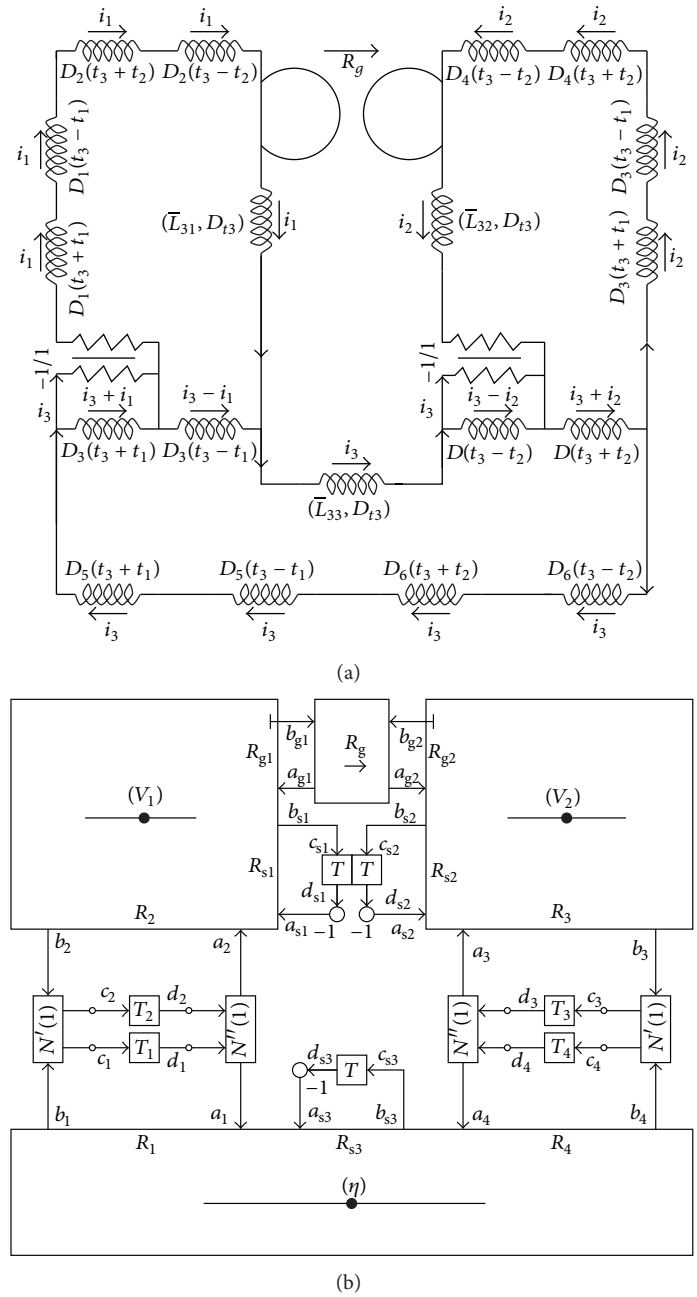


Figure 1: The LSW system representation. (a) MDKC. (b) MDWDF network.

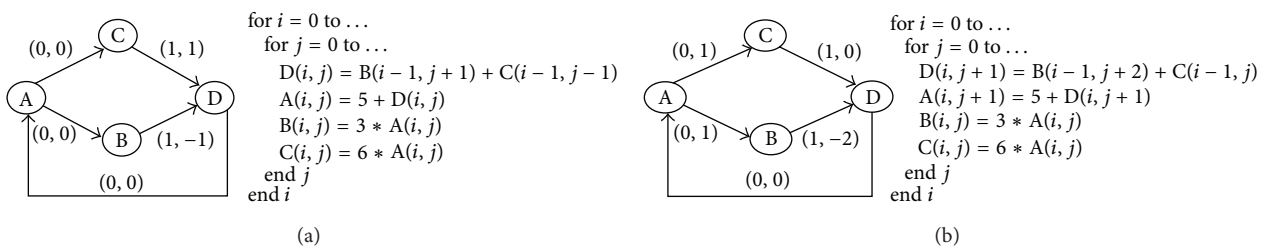


Figure 2: (a) A valid MDFG and its corresponding loop body. (b) A retimed MDFG and its corresponding loop body.

of computation nodes in the loop body, E denotes the set of directed edges representing the dependence between two nodes, d is a function representing the MD delay between two nodes, and t is the discrete time required for computing a certain node [14, 16].

Define a scheduled subspace of a realizable MDFG by $S = \{s : d(e) \cdot s \geq 0, \forall e \in E\}$. The technique of the chained MD retiming [14, 16] for scheduling MD problems starts with finding a schedule vector $s \in S^+$ where S^+ is a strictly positive scheduled subspace defined by

$$S^+ = \{s \in S : d(e) \cdot s > 0, \forall d(e) \neq (0, \ldots, 0), e \in E\}. \quad (2)$$

Given any scheduled element s, a legal retiming function r of a realizable MDFG for each node of a loop body can be obtained when it is orthogonal to s according to [11, 14]. As a consequence, the chained MD scheduling generates a retiming vector $r(u)$ for each computing node u in the MDFG such that the MD delay is pushed from incoming edges of u to its outgoing edges, and new delay of each edge is given by

$$d_r(e_j) = \begin{cases} d(e_j) + r(u) & \text{all outgoing edges } e_j \text{ from } u, \\ d(e_j) - r(u) & \text{all incoming edges } e_j \text{ of } u. \end{cases} \quad (3)$$

It is of importance to note that following the principle of MD retiming [14, 16], $r(u)$ can be obtained immediately without any difficulty by the following formula:

$$r(u) = (K_n - i) \cdot r, \quad (4)$$

where K_n is the maximum length of the existing chains with the highest level number n in the construction process of a MDFG, while i is the level number of the node u. We notice that if an incoming edge possesses zero delay, it is necessary to apply the same retiming function to that incoming node leaving the sum of delays of the loop body unchanged.

Provided the loop body depicted in Figure 3(a) of the MDWDF network, the corresponding MDFG in Figure 3(b) is scheduled by which each operation of the network loop body in Figure 3(b) is executed in one clock cycle by one time unit. As a result, the scheduled table in Table 1(a) is obtained for reference, which clearly shows that the traverse of an iteration of the loop requires a minimum of 7 clock cycles. Choosing the maximum length of the existing chain (D2, E2, F23, G23, H2 or D3, E3, F23, G23, H3) by $K_n = 5$ ((D1, E1, EF1, Gc1, F1, G1, H1 or D4, E4, EF4, Gc2, F4, G4, H4) by $K_n = 7$, resp.) with the highest clock cycle $n = 4$ ($n = 6$, resp.) for the left-loop body (the right-loop body, resp.) of the MDFG of Figure 3(b), (4) yields the retiming vectors listed in Table 2 for each node in the MDFG. Substituting the retiming vector obtained into (3), a full retimed MDFG using the MD push up scheduling is then established as illustrated in Figure 4(a) with its corresponding retimed MDFG loop body in Figure 4(b) and the scheduled table can be traced back as listed in Table 1(b) at each iteration. As compared to the level of clock cycle in Table 1(a), analytical results listed in Table 1(b) have simply demonstrated the achievement of full parallel architecture by the retimed MDFG of Figure 4(a). In view of the retimed MDFG, all edges contain nonzero delay, which eliminates the intraiteration dependencies. Furthermore, the total delays of each loop remains unchanged. To achieve the full parallelism, clearly the left-loop body must require at most 5 parallel processors synchronously executing all operations, while at most 7 parallel processors are required for the right-loop body. The loop body of the retimed MDFG illustrated in Figure 4(b) containing prologue and epilogue processes is executed by P0 to provide the necessary data for the parallel loops, which complementarily completes the process. Thus, provided at most 7 parallel processors named P1, ..., P7, the retimed MDFG has achieved significantly full parallelism by reducing the number of necessary clock cycles into one pass.

Table 1: Schedule tables: (a) MDFG. (b) The retimed MDFG.

(a)

Clock cycle (level number)	Operations			
0	D1	D2	D3	D4
1	E1	E2	E3	E4
2	EF1	F23		EF4
3	Gc1	G23		Gc2
4	F1	H2	H3	F4
5	G1			G4
6	H1			H4
7	C1 C2		C3 C4	
Processor	P0			

(b)

Clock cycle (level number)	Operations							
0	D1	E1	EF1	Gc1	F1	G1	H1	C1
0	D2	E2	F23	G23	H2			C2
0	D3	E3			H3			C3
0	D4	E4	EF4	Gc2	F4	G4	H4	C4
Processor	P1	P2	P3	P4	P5	P6	P7	P0

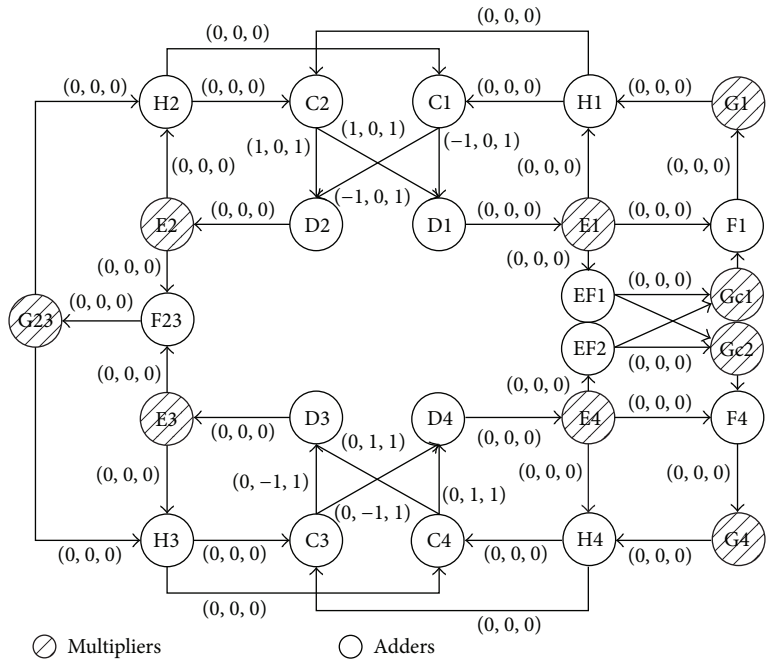
3. Hardware Experiment

In this section, we present empirical simulations for the study of computational efficiency among different MDWDF networks all implemented on the IBM Cell/BE within SONY PS3 with architecture depicted in Figure 5(a) [15]. Comparisons between full parallel and partial/or sequential networks are also given to demonstrate the excellent performance of the proposed full parallel architecture.

3.1. Parallelization across Multiple SPEs and PPE. Before discussing how to make parallelization across multiple SPEs based on the retimed MDFG listed in Table 1(b), we briefly mention some unique features of the IBM Cell/BE

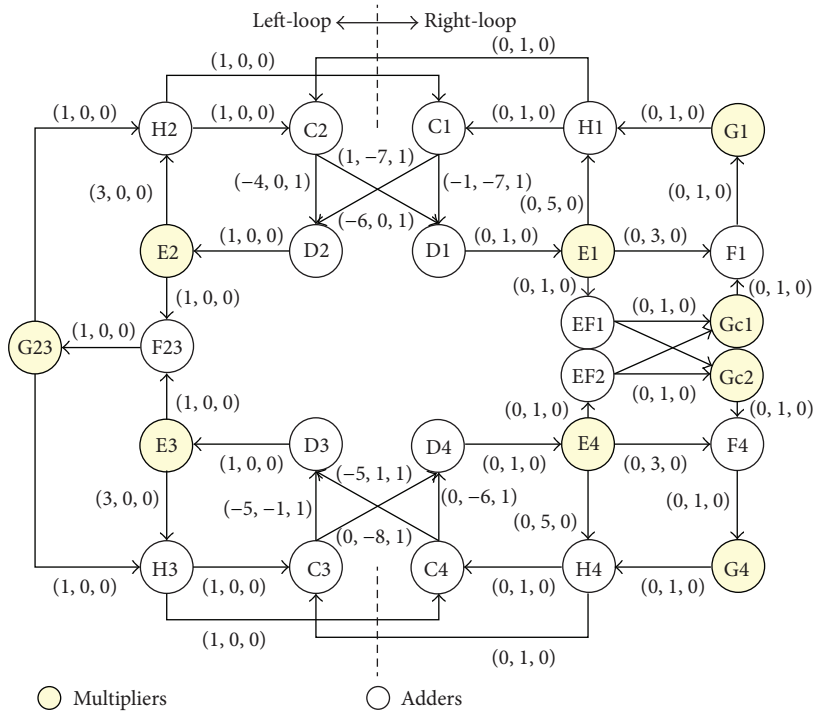
```
for k = 1 to K
 for m = 0 to M
   for n = 0 to N
      D1: ac1(m, n, k) = c1(m + 1, n, k − 1) + c2(m − 1, n, k − 1)
      D2: dc1(m, n, k) = c1(m + 1, n, k − 1) − c2(m − 1, n, k − 1)
      D3: dc2(m, n, k) = c4(m, n − 1, k − 1) − c3(m, n + 1, k − 1)
      D4: ac2(m, n, k) = c4(m, n − 1, k − 1) + c3(m, n + 1, k − 1)
      E1: a1(m, n, k) = −1/2 ∗ ac1(m, n, k)
      E2: a2(m, n, k) = −1/2 ∗ dc1(m, n, k)
      E3: a3(m, n, k) = −1/2 ∗ dc2(m, n, k)
      E4: a4(m, n, k) = −1/2 ∗ ac2(m, n, k)
      EF1: bg1(m, n, k) = −a1(m, n, k)
      EF2: bg2(m, n, k) = −a4(m, n, k)
      Gc1: ag1(m, n, k) = Rg1 ∗ bg1(m, n, k) − Rg2 ∗ bg2(m, n, k)
      Gc2: ag2(m, n, k) = Rg2 ∗ bg1(m, n, k) + Rg1 ∗ bg2(m, n, k)
      F1: SaR1(m, n, k) = a1(m, n, k) + ag1(m, n, k)
      F23: SaL(m, n, k) = a2(m, n, k) + a3(m, n, k)
      F4: SaR2(m, n, k) = a4(m, n, k) + ag2(m, n, k)
      G1: MaR1(m, n, k) = −SaR1(m, n, k)
      G23: MaL(m, n, k) = −SaL(m, n, k)
      G4: MaR2(m, n, k) = −SaR2(m, n, k)
      H1: b1(m, n, k) = a1(m, n, k) + MaR1(m, n, k)
      H2: b2(m, n, k) = a2(m, n, k) + MaL(m, n, k)
      H3: b3(m, n, k) = a3(m, n, k) + MaL(m, n, k)
      H4: b4(m, n, k) = a4(m, n, k) + MaR2(m, n, k)
      C1: c1(m, n, k) = b1(m, n, k) + b2(m, n, k)
      C2: c2(m, n, k) = b1(m, n, k) − b2(m, n, k)
      C3: c3(m, n, k) = b4(m, n, k) − b3(m, n, k)
      C4: c4(m, n, k) = b4(m, n, k) + b3(m, n, k)
   end n
 end m
end k
```

(a)



(b)

Figure 3: (a) The loop body representing the MDWDF network. (b) A MDFG corresponding to the loop body.



(a)

```
for k = 1 to K
  for n = 0 to N % left loop body
P0 —prologue—
    for m = 0 to M − 5
P1  D2: dc1(m + 5, n, k) = c1(m + 6, n, k − 1) − c2(m + 4, n, k − 1)
    D3: dc2(m + 5, n, k) = c4(m + 5, n − 1, k − 1) − c3(m + 5, n + 1, k − 1)
P2  E2: a2(m + 4, n, k) = −1/2 ∗ dc1(m + 4, n, k)
    E3: a3(m + 4, n, k) = −1/2 ∗ dc2(m + 4, n, k)
P3  F23: SaL(m + 3, n, k) = a2(m + 3, n, k) + a3(m + 3, n, k)
P4  G23: MaL(m + 2, n, k) = −SaL(m + 2, n, k)
P5  H2: b2(m + 1, n, k) = a2(m + 1, n, k) + MaL (m + 1, n, k)
    H3: b3(m + 1, n, k) = a3(m + 1, n, k) + MaL (m + 1, n, k)
    end m
P0 —epilogue—
  end n
  for m = 0 to M % right loop body
P0 —prologue—
      for n = 0 to N − 7
P1  D1: ac1(m, n + 7, k) = c1(m + 1, n + 7, k − 1) + c2(m − 1, n + 7, k − 1)
    D4: ac2(m, n + 7, k) = c4(m, n + 6, k − 1) + c3(m, n + 8, k − 1)
P2  E1: a1(m, n + 6, k) = −1/2 ∗ ac1(m, n + 6, k)
    E4: a4(m, n + 6, k) = −1/2 ∗ ac2(m, n + 6, k)
P3  EF1: bg1(m, n + 5, k) = −a1(m, n + 5, k)
    EF2: bg2(m, n + 5, k) = −a4(m, n + 5, k)
P4  Gc1: ag1(m, n + 4, k) = Rg1 ∗ bg1(m, n + 4, k) − Rg2 ∗ bg2(m, n + 4, k)
    Gc2: ag2(m, n + 4, k) = Rg2 ∗ bg1(m, n + 4, k) + Rg1 ∗ bg2(m, n + 4, k)
P5  F1: SaR1(m, n + 3, k) = a1(m, n + 3, k) + ag1(m, n + 3, k)
    F4: SaR2(m, n + 3, k) = a4(m, n + 3, k) + ag2(m, n + 3, k)
P6  G1: MaR1(m, n + 2, k) = −SaR1(m, n + 2, k)
    G4: MaR2(m, n + 2, k) = −SaR2(m, n + 2, k)
P7  H1: b1(m, n + 1, k) = a1(m, n + 1, k) + MaR1(m, n + 1, k)
    H4: b4(m, n + 1, k) = a4(m, n + 1, k) + MaR2(m, n + 1, k)
      end n
P0 —epilogue—
    end m
P0 for m = 0 to M
     for n = 0 to N
       C1: c1(m, n, k) = b1(m, n, k) + b2(m, n, k)
       C2: c2(m, n, k) = b1(m, n, k) − b2(m, n, k)
       C3: c3(m, n, k) = b4(m, n, k) − b3(m, n, k)
       C4: c4(m, n, k) = b4(m, n, k) + b3(m, n, k)
     end n
   end m
end k
```

(b)

Figure 4: (a) A retimed MDFG of the MDWDF network. (b) The corresponding loop body of the retimed MDFG.

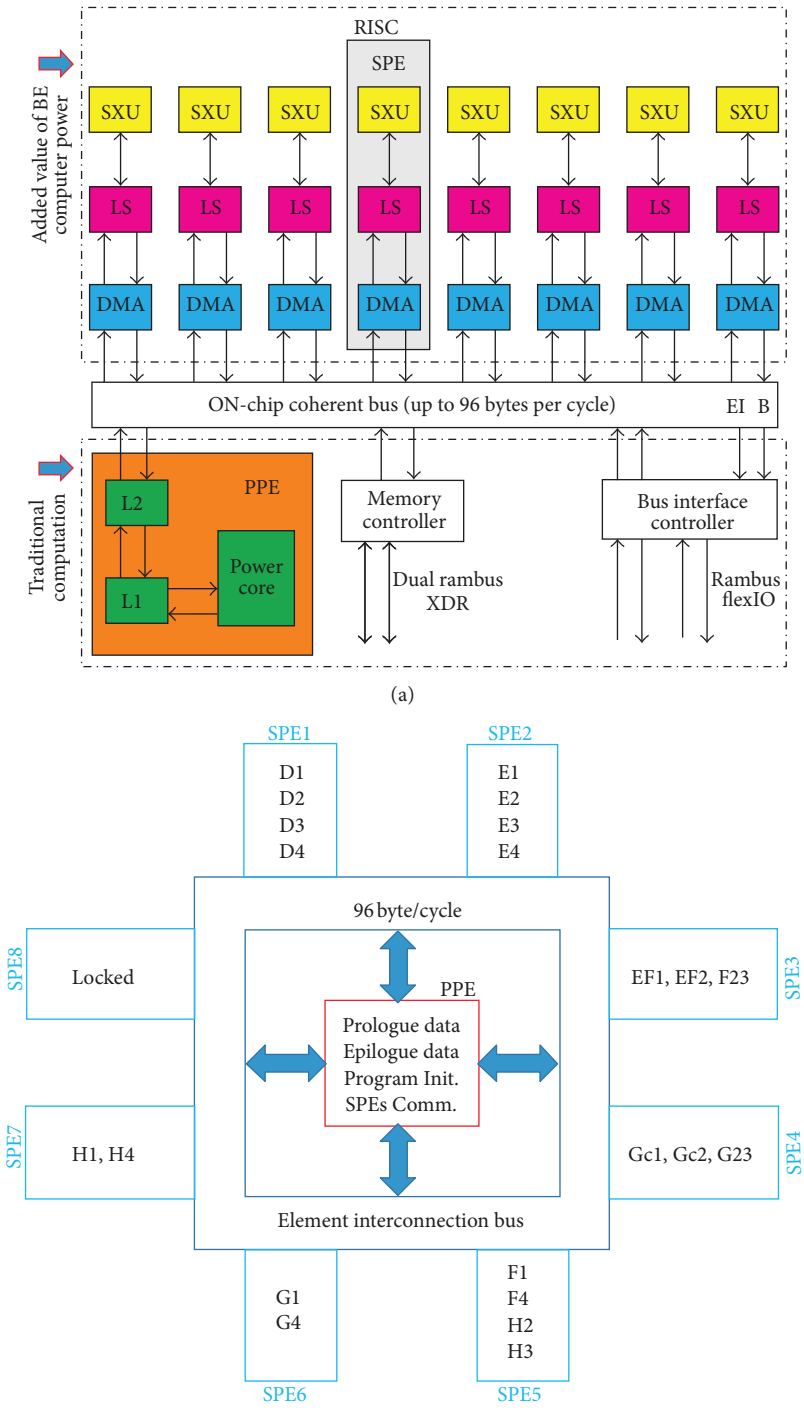


FIGURE 5: (a) IBM cell/BE schema with one PPE and eight SPEs [15]. (b) A custom designed 64 bit power architecture based on the IBM cell/BE for full parallelism of the MDWDF network.

in Figure 5(a). In particular, since the Cell/BE is recognized as a best-of-breed design, which delivers significant computational power, high bandwidth, excellent power and performance, and leading area-saving efficiency within the constraints of a process technology, we generally pinpoint where it possesses some added values of cell broadband engine computer power that is significantly different from the traditional CPU architectures.

In view of Figure 5(a) also according to [15], the most important differences to conventional multicore CPUs is that the Cell/BE is not a homogeneous system with multiple copies of the same core. Instead, the architecture possesses uniquely

Table 2: Retiming vector at each node in the MDFG.

Left-loop body, $r = (1, 0, 0)$, $K_n = 5$		Right-loop body, $r = (0, 1, 0)$, $K_n = 7$	
$r(D2) = (5, 0, 0)$	$r(D3) = (5, 0, 0)$	$r(D1) = (0, 7, 0)$	$r(D4) = (0, 7, 0)$
$r(E2) = (4, 0, 0)$	$r(E3) = (4, 0, 0)$	$r(E1) = (0, 6, 0)$	$r(E4) = (0, 6, 0)$
$r(F2) = (3, 0, 0)$	$r(F3) = (3, 0, 0)$	$r(EF1) = (0, 5, 0)$	$r(EF2) = (0, 5, 0)$
$r(G2) = (2, 0, 0)$	$r(G3) = (2, 0, 0)$	$r(Gc1) = (0, 4, 0)$	$r(Gc2) = (0, 4, 0)$
$r(H2) = (1, 0, 0)$	$r(H3) = (1, 0, 0)$	$r(F1) = (0, 3, 0)$	$r(F4) = (0, 3, 0)$
		$r(G1) = (0, 2, 0)$	$r(G4) = (0, 2, 0)$
		$r(H1) = (0, 1, 0)$	$r(H4) = (0, 1, 0)$
$r(C1) = r(C2) = r(C3) = r(C4) = (0, 0, 0)$			

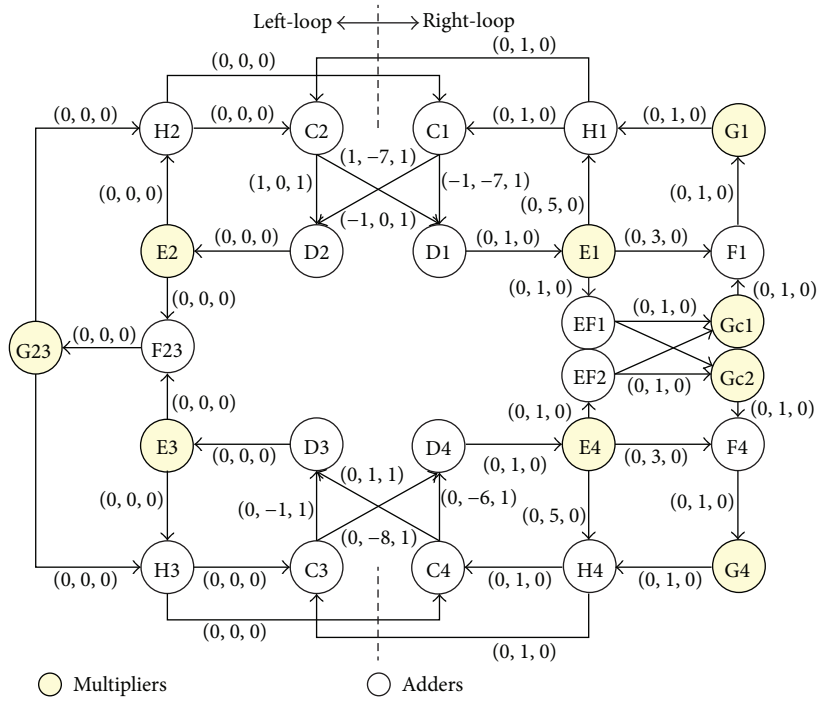
a heterogeneous system with each cell strategically designed as a reduced instruction set computer (RISC), which consists of a 64-bit PPE core and eight SPEs coprocessor running independently. Furthermore, each SPE contains 256 KB local store, a memory flow controller (MFC), and a synergistic processing unit (SPU) with a single instruction multiple data (SIMD) processing unit and 128 registers of 128 bits each. In addition, the architecture holds an element interconnection bus (EIB) with an internal bandwidth of more than 300 GB/s (per 3.2 GHz Cell processor), responsible of controlling data transfer between the SPEs. The maximum bandwidth from the SPEs to main memory can be geared up to 25 GB/s.

Since a homogeneous programming model is adopted in the programming and therefore has no individual SPE-to-SPE communication, from a programmer's perspective it makes no difference for which these SPEs are physically located. In fact, keeping all 7 SPEs fully utilized (1 SPE is basically locked by SONY) requires efficient load balancing. We, thus, follow the standard approach of defining the SPE working tasks by subdividing the retimed MDFG operations into a set of operations as listed in Table 1(b). More specifically, with the help of memory flow controller-direct memory access (MFC-DMA) queue installed in each SPE, the 8 processors, P0, . . . , P7, scheduled in the loop body of the retimed MDFG in Figure 4(b) can be assigned to the Cell/BE as PPE, SPE1,. . ., SPE7, respectively. As a result, the full parallelism in the left-loop body renders 5 SPEs synchronously, that is, SPE1,. . ., SPE5, while the right-loop body is tackled by 7 SPEs, that is, SPE1,. . ., SPE7, all running independently. From this shared task queue, each SPE dynamically fetches a new set of MDFG operations and renders it. As accesses to this task queue must be synchronized, we employ the Cell's atomic lookup and update capabilities meaning that an integer variable specifying the ID of the next set to be rendered must be allocated in system memory. Since this variable is visible among all SPEs, each time when a SPE queries the value of the variable, it performs an atomic fetch-and-increment instruction. This atomic update mechanism allows all SPEs to be working fully independently from both other SPEs and PPE, requiring no communication among those units. Figure 5(b) shows the efficiency of the dynamic load balancing for the fully parallelized model based on the retimed MDFG of Figure 4(a) to carry out 8 processors (1PPE + 7SPEs), while the nonparallel model based on Figure 3(a) is implemented only by the PPE core itself. Apart from the parallel process implemented on the SPE coprocessors, there are processes of prologue, epilogue, and program initiation, which are controlled fully by the PPE core.

Table 3: Maximum CPU runtime for different types of model.

Maxi. CPU Time (sec)		Field domain: 200×200 (m^2) Scheduled Time Step (group)			
Grids	Model Type	25 (G_1)	50 (G_2)	100 (G_3)	200 (G_4)
51×51	Full sequential	15.75 (4.15×)	32.33 (4.35×)	60.77 (4.14×)	121.3 (3.91×)
	Left-loop parallel	13.71 (3.61×)	28.21 (3.79×)	56.75 (3.86×)	117.7 (3.79×)
	Right-loop parallel	12.93 (3.41×)	26.26 (3.52×)	52.76 (3.59×)	110.25 (3.56×)
	Full parallel	3.79	7.44	14.69	31.01
201×201	Full sequential	2931 (4.14×)	5337 (3.77×)	10234 (3.72×)	20622 (3.85×)
	Left-loop parallel	4595 (6.49×)	9141 (6.46×)	17273 (6.27×)	37791 (7.05×)
	Right-loop parallel	4053 (5.73×)	8465 (5.98×)	16813 (6.11×)	35026 (6.53×)
	Full parallel	707	1415	2753	5356

3.2. Simulation Results and Performance between SPEs. In this subsection, the CPU runtime of 4 different MDWDF network models with the scheduled computational grids (51×51 and 201×201) has been put on test based on 4 groups of temporal point for implementation: G_k, $k = 1, \ldots, 4$. These models include the full sequential (Figure 3), full parallel (Figure 4), right-loop body parallel (Figure 6), and left-loop body parallel (Figure 7) MDWDF networks. As each group implements a specified discrete temporal point at $t_k = 25 \times 2^{k-1}$, the temporal point for implementation is increased by 2 times between groups. Figure 8(a) shows results of the full sequential model with grids 51×51, which has clearly illustrated that its CPU runtime has nearly matched the increment of its corresponding temporal point as also indicated in the first-half part of Table 3. To improve the network's performance and make comparison with other partial parallelism models, the computational efficiency based on

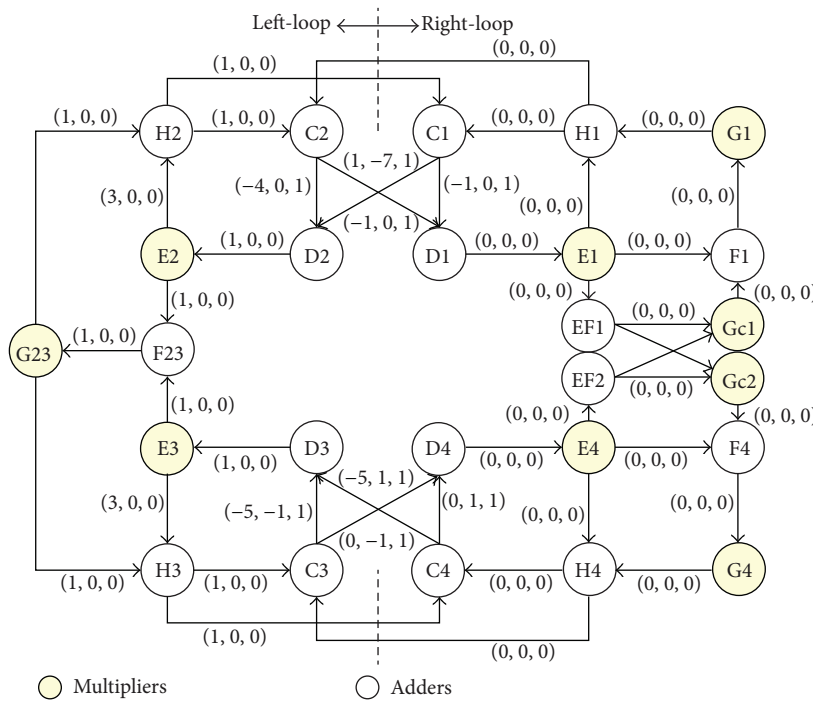


(a)

```
for k = 1 to K
  for n = 0 to N % left loop body
    for m = 0 to M
      D2: dc2(m, n, k) = c1(m + 1 , n, k − 1) − c2(m − 1, n, k − 1)
      D3: dc2(m, n, k) = c4(m, n − 1, k − 1) − c3(m, n + 1, k − 1)
      E2: a2(m, n, k) = −1/2 ∗ dc1(m, n, k)
      E3: a3(m, n, k) = −1/2 ∗ dc2(m, n, k)
      F23: SaL(m, n, k) = a2(m, n, k) + a3(m, n, k)
      G23: MaL(m, n, k) = −SaL(m, n, k)
      H2: b2(m, n, k) = a2(m, n, k) + MaL (m, n, k)
      H3: b3(m, n, k) = a3(m, n, k) + MaL (m, n, k)
    end m
  end n
    for m = 0 to M % right loop body
  P0 —prologue—
      for n = 0 to N − 7
  P1  D1: ac1(m, n + 7, k) = c1(m + 1, n + 7, k − 1) + c2(m − 1, n + 7, k − 1)
      D4: ac2(m, n + 7, k) = c4(m + 1, n + 6, k − 1) + c3(m, n + 8, k − 1)
  P2  E1: a1(m, n + 6, k) = −1/2 ∗ ac1(m, n + 6, k)
      E4: a4(m, n + 6, k) = −1/2 ∗ ac2(m, n + 6, k)
  P3  EF1: bg1(m, n + 5, k) = −a1(m, n + 5, k)
      EF2: bg2(m, n + 5, k) = −a4(m, n + 5, k)
  P4  Gc1: ag1(m, n + 4, k) = Rg1 ∗ bg1(m, n + 4, k) − Rg2 ∗ bg2(m, n + 4, k)
      Gc2: ag2(m, n + 4, k) = Rg2 ∗ bg1(m, n + 4, k) + Rg1 ∗ bg2(m, n + 4, k)
  P5  F1: SaR1(m, n + 3, k) = a1(m, n + 3, k) + ag1(m, n + 3, k)
      F4: SaR2(m, n + 3, k) = a4(m, n + 3, k) + ag2(m, n + 3, k)
  P6  G1: MaR1(m, n + 2, k) = −SaR1(m, n + 2, k)
      G4: MaR2(m, n + 2, k) = −SaR2(m, n + 2, k)
  P7  H1: b1(m, n + 1, k) = a1(m, n + 1, k) + MaR1(m, n + 1, k)
      H4: b4(m, n + 1, k) = a4(m, n + 1, k) + MaR2(m, n + 1, k)
      end n
  P0 —epilogue—
    end m
  for m = 0 to M
    for n = 0 to N
      C1: c1(m, n, k) = b1(m, n, k) + b2(m, n, k)
      C2: c2(m, n, k) = b1(m, n, k) − b2(m, n, k)
      C3: c3(m, n, k) = b4(m, n, k) − b3(m, n, k)
      C4: c4(m, n, k) = b4(m, n, k) + b3(m, n, k)
    end n
  end m
end k
```

(b)

Figure 6: (a) The right-loop body retimed MDFG of the MDWDF network. (b) The corresponding loop body of the right-loop body retimed MDFG.



(a)

```
for k = 1 to K
  for n = 0 to N % left loop body
 P0 —prologue—
    for m = 0 to M − 5
P1  D2: dc2(m + 5, n, k) = c1(m + 6 , n, k − 1) − c2(m + 4, n, k − 1)
    D3: dc2(m + 5, n, k) = c4(m + 5, n − 1, k − 1) − c3(m + 5, n + 1, k − 1)
P2  E2: a2(m + 4, n, k) = −1/2 ∗ dc1(m + 4, n, k)
    E3: a3(m + 4, n, k) = −1/2 ∗ dc2(m + 4, n, k)
P3 → F23: SaL(m + 3, n, k) = a2(m + 3, n, k) + a3(m + 3, n, k)
P4 → G23: MaL(m + 2, n, k) = −SaL(m + 2, n, k)
P5  H2: b2(m + 1, n, k) = a2(m + 1, n, k) + MaL (m + 1, n, k)
    H3: b3(m + 1, n, k) = a3(m + 1, n, k) + MaL (m + 1, n, k)
    end m
 P0 —epilogue—
  end n
for m = 0 to M % right loop body
  for n = 0 to N
    D1: ac1(m, n, k) = c1(m + 1, n, k − 1) + c2(m − 1, n, k − 1)
    D4: ac2(m, n, k) = c4(m, n − 1, k − 1) + c3(m, n + 1, k − 1)
    E1: a1(m, n, k) = −1/2 ∗ ac1(m, n, k)
    E4: a4(m, n, k) = −1/2 ∗ ac2(m, n, k)
    EF1: bg1(m, n, k) = −a1(m, n, k)
    EF2: bg2(m, n, k) = −a4(m, n, k)
    Gc1: ag1(m, n, k) = Rg1 ∗ bg1(m, n, k) − Rg2 ∗ bg2(m, n, k)
    Gc2: ag2(m, n, k) = Rg2 ∗ bg1(m, n, k) + Rg1 ∗ bg2(m, n, k)
    F1: SaR1(m, n, k) = a1(m, n, k) + ag1(m, n, k)
    F4: SaR2(m, n, k) = a4(m, n, k) + ag2(m, n, k)
    G1: MaR1(m, n, k) =−SaR1(m, n, k)
    G4: MaR2(m, n, k) =−SaR2(m, n, k)
    H1: b1(m, n, k) = a1(m, n, k) + MaR1(m, n, k)
    H4: b4(m, n, k) = a4(m, n, k) + MaR2(m, n, k)
  end n
end m
for m = 0 to M
  for n = 0 to N
    C1: c1(m, n, k) = b1(m, n, k) + b2(m, n, k)
    C2: c2(m, n, k) = b1(m, n, k) − b2(m, n, k)
    C3: c3(m, n, k) = b4(m, n, k) − b3(m, n, k)
    C4: c4(m, n, k) = b4(m, n, k) + b3(m, n, k)
  end n
end m
end k
```

(b)

Figure 7: (a) The left-loop body retimed MDFG of the MDWDF network. (b) The corresponding loop body of the left-loop body retimed MDFG.

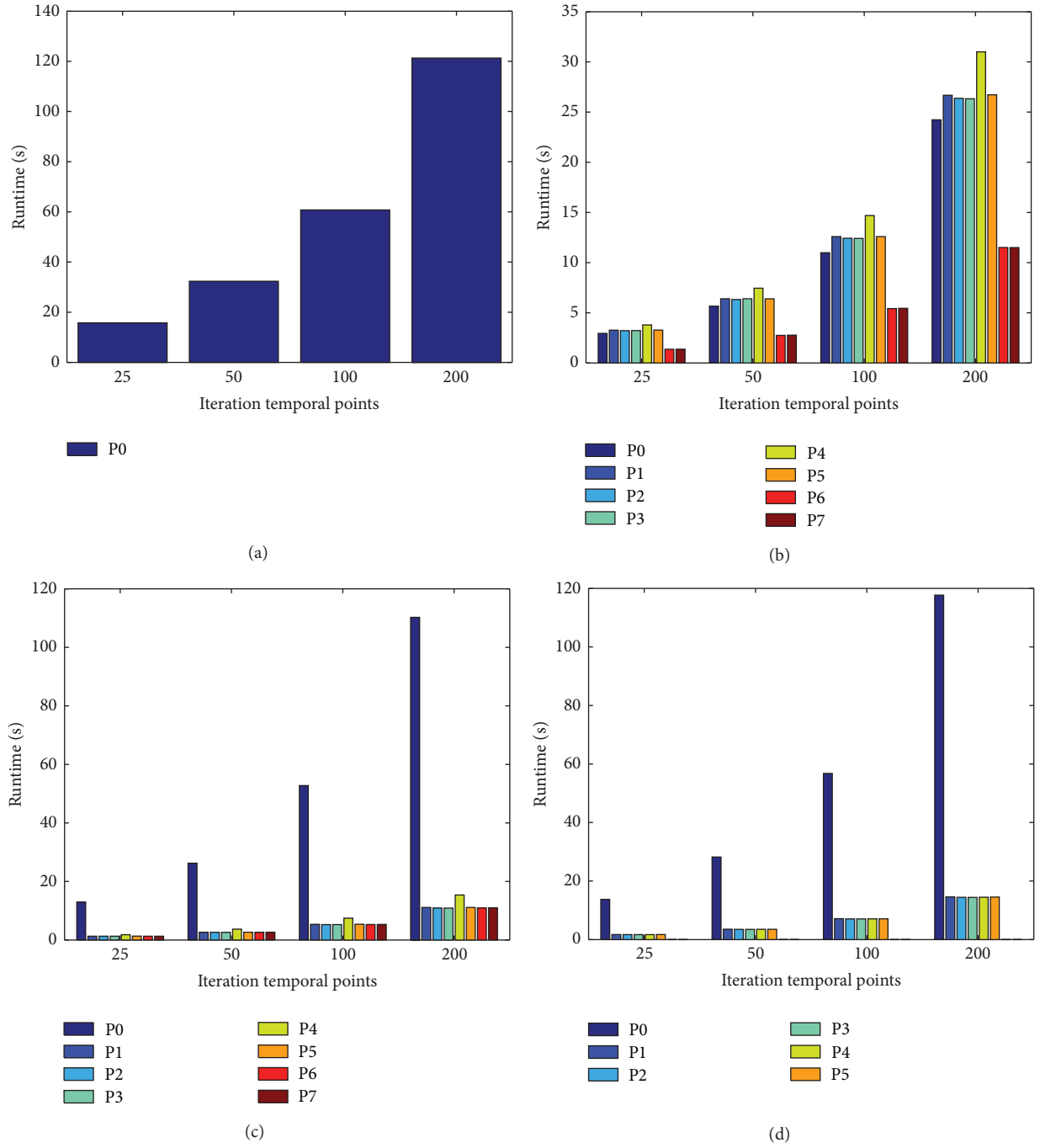


Figure 8: Performance comparison between four models of MDWDF network with scheduled grids 51 × 51. (a) Full sequential model. (b) Full parallel model. (c) Right-loop body retimed model. (d) Left-loop body retimed model.

the MDFG of Figures 4(a), 6(a), and 7(a) are presented in Figures 8(b)–8(d) for full parallelism model, the right-loop retimed body, and the left-loop retimed body, respectively. Figure 8(b) shows that the full parallel model receives a significant improvement with up to 4.35× more efficiency than the full sequential one based on details given in Table 3.

Similar advantages of using the full parallel model are also applied, which outperforms the right-loop and left-loop retimed models. As can be seen from Figures 8(c) and 8(d), the full parallel model has gained up to 3.59× and 3.86× faster than the right-loop and left-loop retimed models, respectively. The cause of less efficiency can be found in Figure 6(b) where the left-loop body, although it technically requests 5 SPEs according to the scheduled retimed MDFG in Table 2, is implemented sequentially only by the PPE core resulting in a huge burden of 5× average CPU runtime of

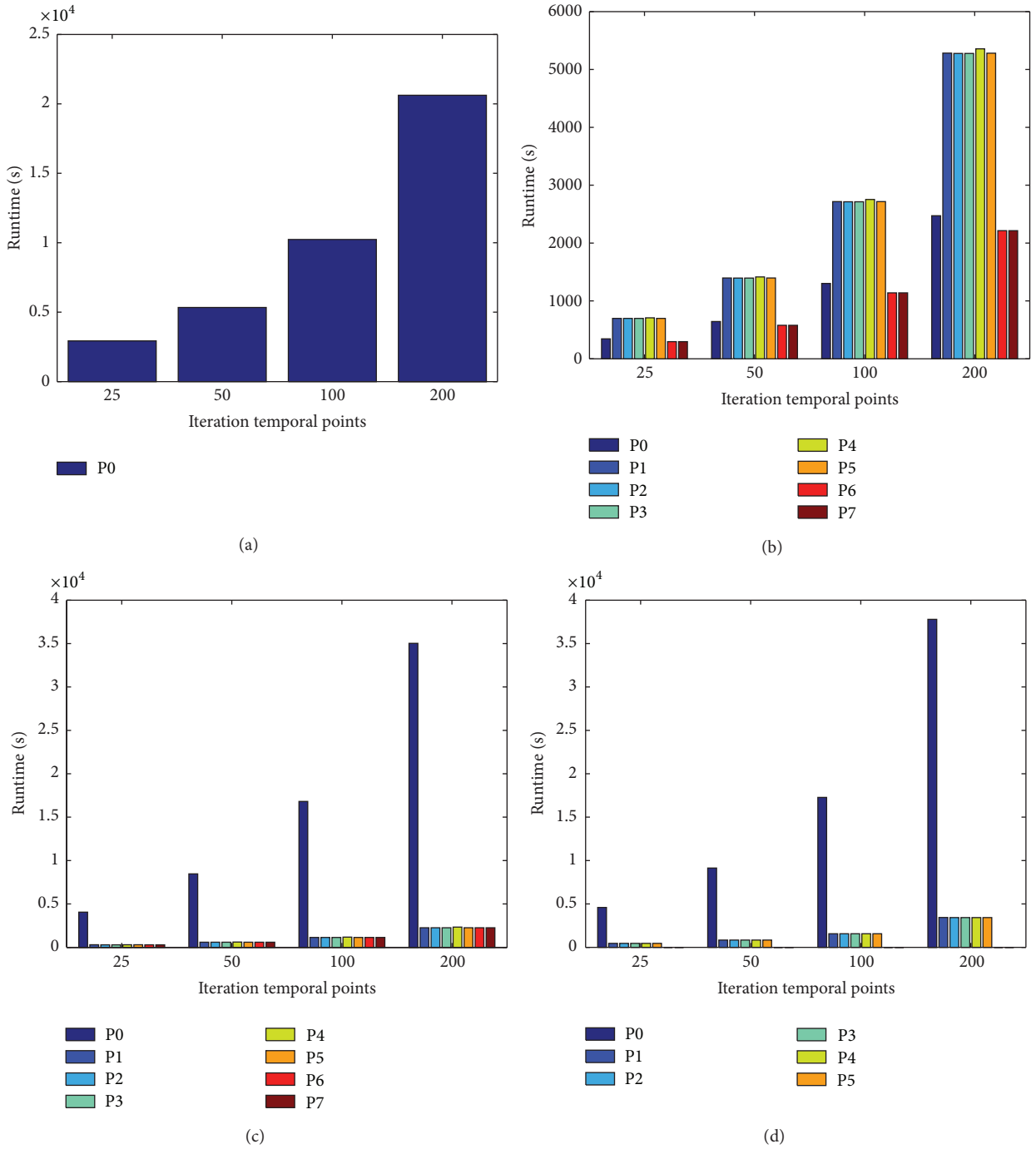


Figure 9: Performance comparison between four models of MDWDF network with scheduled grids 201 × 201. (a) Full sequential model. (b) Full parallel model. (c) Right-loop body retimed model. (d) Left-loop body retimed model.

SPEs being added to the PPE core. Whereas for the left-loop retimed body, Figure 7(b) indicates that the right-loop body is sequentially performed by the PPE core encountering 7× the average CPU workload of SPEs being added on the PPE core.

Having the computational complexity increased 16 times by arranging the grids of 201 × 201, Figures 9(a)–9(d) clearly demonstrate that the full parallel architecture of MDWDF network has achieved its goal by significantly boosting the performance of nonparallel MDWDF network. More specifically, the full parallel model outperforms models of sequential and partial parallelism of right/left-loop bodies by the least 3.72×, 5.73×, and 6.27× runtimes, respectively, whose detail is listed in the second-half part of Table 3. More specifically, we look at Table 3 concerning about the performance of full parallel model against that of the full sequential one for two scheduled grids. Despite the increase of computational complexity, the full parallel model still

remains at the range of gain [3×, 4×] over the full sequential model. By contrast, the full parallel model receives a bigger range of gains with [3×, 6×] and [3×, 7×] over the right- and left-loop retimed bodies, respectively. The increment of gain over these partial parallel models may be attributed to improper allocation of massive data transfer between the PPE core and these SPEs coprocessors. These results, nevertheless, have clearly suggested that the scheduled 64-bit power architecture for full parallelism of the MDWDF network model provides the key performance advantage by fully equipping the 8 decoupled SPE SIMD engines (1PPE + 7SPEs) with dedicated resources including large register files and DMA channels.

4. Conclusion

In this study, we have studied the significant computer runtime performance for different models of MDWDF network implemented on the IBM cell/BE. The full parallelism of the MDWDF network was carried out by working together with 8 decoupled SPU SIMD engines, each with dedicated resources including DMA channels. In particular of interest, a custom design of 64-bit power architecture was developed, which facilitates the PPE core processor to control the prologue, epilogue, and program initiation, while the massive parallel processes of the right- and left-loop bodies were performed by at most 7 SPE coprocessors. This has rendered the Cell/BE to utilize 8 times more SIMD capability (for up to 16-way data parallelism) than the conventional processors with vector architecture extensions, such as the PPC970 in the G5. Simulation results have demonstrated that the CPU runtime speed of the full parallel model outperforms that of the other 3 models by up to 4× improvement for scheduled grids 51 × 51; while for scheduled grids 201 × 201, the full parallel model can significantly gain by up to 7× improvement.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

References

[1] P. K. Kundu, *Fluid Mechanics*, Academic Press, London, UK, 1990.

[2] Z. Kowalik and T. S. Murty, *Numerical Modelling of Ocean Dynamics*, World Scientific Publishing, 1993.

[3] A. Fettweis, "Wave digital filters: theory and practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, 1986.

[4] A. Fettweis and G. Nitsche, "Transformation approach to numerically integrating PDEs by means of WDF principles," *Multidimensional Systems and Signal Processing*, vol. 2, no. 2, pp. 127–159, 1991.

[5] A. Fettweis and G. Nitsche, "Numerical integration of partial differential equations using principles of multidimensional wave digital filters," *Journal of VLSI Signal Processing*, vol. 3, no. 1-2, pp. 7–24, 1991.

[6] J. C.-H. Tseng, N. D.-T. Dao, and C.-C. Chang, "Modeling and visualizing seismic wave propagation in elastic medium using multi-dimension wave digital filtering approach," *World Academy of Science, Engineering and Technology*, vol. 69, pp. 424–430, 2010.

[7] H. Krauß, R. Rabenstein, and M. Gerken, "Simulation of wave propagation by multidimensional digital filters," *Simulation Practice and Theory*, vol. 4, no. 6, pp. 361–382, 1996.

[8] C. H. Tseng and S. S. Lawson, "Discrete modelling of shallow water equations using the multidimensional wave digital filters," in *Proceedings of the European Conference in Circuit Theory and Design*, Krakow, Poland, 2003.

[9] C.-H. Tseng and S. Lawson, "Initial and boundary conditions in multidimensional wave digital filter algorithms for plate vibration," *IEEE Transactions on Circuits and Systems I*, vol. 51, no. 8, pp. 1648–1663, 2004.

[10] C. H. Tseng, "Modelling and visualization of a time-dependent shallow water system using nonlinear Kirchhoff circuit," *IEEE Transactions on Circuits and Systems*, vol. 59, no. 6, pp. 1265–1277, 2012.

[11] C.-H. Tseng and S. Lawson, "Full parallel process for multidimensional wave digital filtering via multidimensional retiming technique," in *Proceedings of the IEEE International Symposium on Cirquits and Systems (ISCAS '04)*, pp. III209–III212, Vancouver, Canada, May 2004.

[12] C. H. Tseng, "Numerical stability verification of a two-dimensional time-dependent nonlinear shallow water system using multidimensional wave digital filtering network," *Circuits, Systems and Signal Processing*, vol. 32, no. 1, pp. 299–319, 2013.

[13] C. H. Tseng and Y. L. Jeang, *FPGA Digital IC Design and Practice: Use Verilog HDL and Xilinx ISE*, Tsang Hai Book Publishing Co., 1st edition, 2012 (Chinese).

[14] N. L. Passos and E. H.-M. Sha, "Achieving full parallelism using multidimensional retiming," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 11, pp. 1150–1163, 1996.

[15] B. Flachs, S. Asano, S. H. Dhong et al., "A streaming processing unit for a CELL processor," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '05)*, pp. 134–135, February 2005.

[16] N. L. Passos and E. Sha, "Full parallelism in uniform nested loops using multidimensional retiming," in *Proceedings of the International Conference on Parallel Processing*, pp. 130–133, Saint Charles, Ill, USA, 1994.