Turkish Journal of Computer and Mathematics Education *Vol.12 No.4 (2021), 1400-1413*

Research Article

CGRA MODULO SCHEDULING FOR ACHIEVING BETTER PERFORMANCE AND INCREASED EFFICIENCY

Siva Sankara Phani.T [a], M.Durga Prakash[b]

[a,] *Department of ECE, Koneru Lakshmaiah Education Foundation,Green Fields,Vaddeswaram,Guntur 522502,AP,India*
[b]*Department of ECE, V R Siddhartha Engineering College(Autonomous),Kanuru,Vijayawada 520007,AP,India*

Article History: Received: 11 January 2021; Accepted: 27 February 2021; Published online: 5 April 2021

Abstract: Coarse-Grained Reconfigurable Architectures (CGRA) is an effective solution for speeding up computer-intensive activities due to its high energy efficiency and flexibility sacrifices. The timely implementation of CGRA loops was one of the hardest problems in the analysis. Modulo scheduling (MS) was productive in order to implement loops on CGRAs. The problem remains with current MS algorithms, namely to map large and irregular circuits to CGRAs over a fair period of compilation with restricted computational and high-performance routing tools. This is mainly due to an absence of awareness of major mapping limits and a time consuming approach to solving temporary and space-related mapping using CGRA buffer tools. It aims to boost the performance and robust compilation of the CGRA modulo planning algorithm. The problem with the CGRA MS is divided into time and space and the mechanisms between the two problems have to be reorganized. We have a detailed, systematic mapping fluid that addresses the algorithms of the time mapping problem with a powerful buffer algorithm and efficient connection and calculation limitations. We create a fast-stable algorithm for spatial mapping with a retransmission and rearrangement mechanism. With higher performance and quicker build-up time, our MS algorithm can map loops to CBGRA. The results show that, given the same compilation budget, our mapping algorithm results in a better rate for compilation. The performance of this method will be increased from 5% to 14%, better than the standard CGRA mapping algorithms available.

Keywords:CGRA, Modulo scheduling, Spatial mapping,Temporal mapping.

1.Introduction

CGRA have become an important subject for academia and industry in recent years. Our consciousness reveals that at least 40 CGRAs have been engineered to accommodate various applications in recent decades. These CGRAs would boost mobile device performances [1]. They are intended to increase the output of mobile apps [2]. Thus, the development of the CGRA software environment is difficult because of the different features and implementations of the CGRA hardware architecture. Many compilers of CGRA are computer-based [3].

Thus, in compilers, the generality and programmability of the formulation of consistent computational issues is essential to a generalized CGRA model. ADRES [4], CRC [5] or CGRA-ME [6] for simplified CGRA modelling template is representative projects of CGres loop accelerators. These models describe the core components of the CGRA architecture, which includes an RPU, with one or more arrays, hierarchical buffer and network interfaces. A CGRA flavour-based template model can be parameterized for components within a CGRA design template. The compiler which was extracted from the software application in this generalized CGRA prototype can formulate a consistent Data Flow Graph (DFG) problem and its map ping object is subject to the CGRA execution scheme.

The software pipes CGRA internal loops on typical running systems overlaying the following iterations of the loop [7]. Work is carried out simultaneously in the same and multiple variations, and sequencing and parallel can be used with the instructions. The interval between the two subsequent loop iterations is called the starting interval (II). The smaller the better is the key tool for measuring mapping performance! In general, compilers use modular CGRA loops that have proven to be an NP mapping problem.

The CGRA Module Scheduling (MS) problem was posed before that work [8]. DFG has been mapped into a space-times chart that models the system resources and routing capability of CGRAs and is intended to minimise levels of II in the abstract 3D-CGRA extension module. Multiple methods, including time assignment, PE positioning, PE routing and tampon assignment, include mapping problems (registers). Models in Hamzeh [9] are listed as integrating or decomposing policies. Node by node, the time allocation, buffer assignment and PE positioning and routing method have been implemented via an integrated mapping strategy. Each technique can be formulated into a single problem with a particular goal which is distinct from others rather than handled in a decomposed mapping strategy separately. Two simple decomposed processes are described conceptually as time mapping for any mapping method based on current mapping decomposition or mapping. In this paper two simple decomposed processes can be established conceptually.

2. Literature Survey

Existing CGRA MS algorithms with different mapping policies:

(1) Both policy areas are under-used for buffer resources: most CG RA projects have no algorithm to optimise buffer resources of any kind. In some ventures, for instance, REGIMap[10] uses Local Register Buffers in Pes[11] (OMB). Whereas, for both LRB and OMB[12], the RAMP is only used separately. Where CGRA's

buffer resources are insufficient, competitive efficiency includes the combining use of PE, Global Register Buffer (GRB), LRB and OMB.
(2) Good efficiency and long algorithm compilation based on the integrated mapping policies: while classic mapping performance in realistic mapping process budgets such as simulated renewal[13] optimizations of the particulate swarms[14] or linear integer(ILP) programming can be achieved.
(3) Opt out of the performance of the algorithm easily with integrated mapping policy: other heuristics, including EMS[15], GraphMinor[16], are protected by certain special aspects during the mapping process, but not generalized.
(4) The interconnection and the restriction of machine resources with decomposed mapping algorithms have proved successful. In spite of this, the algorithm for spatial mapping is ineffectually researched when time mapping doesn't realize that CGRA's inadequate relation and routing resources contribute to a spatial mapping failure.
(5) The current mapping rules like EPIMap[17], REGIMap, Conflict-free[18], MemAware and RAMP should be included in space-map ping buffering, PE placement as well as space maps. They formulate a problem of spatial mapping that is also an NP concern. If problem 4 occurs, a long-term approach is decided for spatial mapping, which leads to loss of time.
There are reasons why the new mapping approach is being considered to address the CGRA MS problem more robustly and efficiently. This paper sums up the contribution as follows:
1. We examine current CGRA algorithms of mapping and suggest a mapping theory breaking down space and time charts. For greater efficiency and robust compilation we argue that time mapping should be more attentive than space mapping. We therefore create a systemic time map flow, which carefully analyses and rearranges the DFG for spatial mapping output and code generation.
2. In time-mapping we cause a buffer allocation problem by setting restrictions and rules for the different buffer sources and assigning routing paths to optimise the issued device resources according to their respective restrictions and rules. With the same hybrid buffer, we address the issue with a buffer that heuristically assigns buffer values in life. Therefore, our algorithm adapts extremely well with minimum buffer resources over CGRAs.
3. During time mapping, we also suggest heuristics to prevent spatial mapping incapacity and rearrange operations in the DFG to ensure that the root mapping process is conducted as follows.
4. We propose a quick and efficient Spatial Mapping System which combines spatial mapping mechanisms to ensure effective recovery and reorganization. In the future, with a greedy-based algorithm we can easily produce optimal maps for minimum II. Failure can trigger recovery and rearrangement algorithm to have been successfully mapped.
Experiments demonstrate our mapping technique to ensure the stability of the time; in a given time budget all selected kernels of the loop are mapped. In comparison with advanced CGRA Mapping algorithms, our mapping processes increase the efficiency of successfully mapped loops from 5.4% to 14.2% and the build time between X24 and X1089 is on average faster.

2.1 CGRA Compilation

With loop extraction, the CGRA compilation begins and converts the code areas into the DFG framework. The DFG is composed of a number of V and E nodes, which are the valid functions of any V 2 V node, including arithmetical, logical or memory access. A mapping relation between DFG and a CGRA resource chart is used in the CGRA modulo planning technique.
A lot of CGRA modular heuristics have existed since 2002. These features, including map organisational pattern, buffer methods specifically used to optimise mapping processes, each mapping function, it's positioning and routing system and the 3D-CGRA models you select, are summarised in heuristic terms in Table 1. The aforementioned integrated and decomposed mapping methodology analyses these mappings.
Table 1: Mappings Comparison(P&R: Placement and Routing; MRRG: Modulo Routing Resource Graph;MRT: Modulo Reservation Table; TEC: Time Extended CGRA)

Mapping	Pattern	Leveragedbuffers	Distinguishedfeatures	P&Rmethod	3D-CGRAforma
DRESC	Integrated	PE+LRB	Simulatedannealing(SA)	CostFunction	MRRG
PSOMap	Integrated	PE+LRB	Particleswarmoptimization	CostFunction	MRRG
AA-ILP	Integrated	PE+LRB	Integer LinearProgramming(ILP)	ILPSolver	MRRG
MGE	Integrated	PE+LRB	Skewing theschedulingspace	CostFunction	MRT

EMS	Integrated	PE+LRB	Edge-centric	CostFunction	MRT
RA	Integrated	PE+LRB	Backtracking	CostFunction	MRRG
GM	Integrated	PE	Graph-minor+Routingpathssharing	CostFunction	TEC
Bimodal	Integrated	PE+LRB+CU	Adaptivecostfunction	CostFunction	MRRG
SPR	Decomposed	PE+LRB	Simulatedannealing(SA)	CostFunction	MRRG
EPIMap	Decomposed	PE	Epimorphism+Re-compute	MaxCliqueFinding	TEC
REGIMap	Decomposed	PE+LRB	LRBaware+Reordering	MaxCliqueFinding	TEC
MA	Decomposed	PE+LRB+OMB	OMBaware	MaxCliqueFinding	TEC
RAMP	Decomposed	PE+LRB+OMB	Re-schedule	MaxCliqueFinding	TEC
Our Method	Decomposed	PE+LRB+GRB+OMB	Allbuffersaware	CostFunction	TEC

3. Integrated Mapping

Pioneers in solving CGRA MS are the classic schemes for heuristic optimization, for example the simulation of ring, the optimization of particle swarms, and the integrated linear programming. The simulated annealing (SA), for example, is used by DRESC. As a social representation for birds PSOMap[14] uses particle swarm optimization. At the start of these papers the issue of mapping DFG to the 3D Routing Resource Graph (MRRG). Differences in time and resources are altered in each DFG procedure and conflicts in resources are controlled.

The conflict over resources can be considered to be multiple operations conducted within the same PE and cycle. This is achieved by the compilers until they have a good map ping and cannot further enhance (converge) performance or until the mapping completes the time budget to compile.The DRESC and PSOMap are used for search operations in order to combine time and space mapping, positioning and routing. When the local registry buffer is reduced, a last priority buffer allocation for both map pings occurs. It's a big overhead substitute.

For the mapping problem of DFG and MRRG, maps such as [19] and [20] uses the 0-1 integer linear programming. Their objective is to evaluate the specified value for the objective function of all variables. Machines and routing resources in the mapping of large DFGs to CGRA are also compiled relatively long by the ILP map pings. This explains why it is important to find the best possible solution for the power of the linear programming solver. With the number of variables and constraints, the time complexity of the ILP solver increases exponentially. For this reason ILP-based mapping for large loops is always difficult. For eg, two kernels, according to, are compiled over 24 hours and have no more than 30 DFG operating numbers. The use of ILP can be incredibly long at our DFG Research Centre, with a cumulative running number of 154.

The value of interoperable routing is anticipated in the EMS. Instead of a node focus approach which prioritizes PE placement, EMS adopts an edge focused approach that concentrates primarily on routing. When the mapping is not done, the EMS lacks the recuperation or restoration feature, which results in a higher performance rate. Bimodal's planner enhances the EMS to increase time intercompatibility by retrotracking and priority and cost feature adaptations on a wide variety of circuits.

The diagramming of every system by means of costing features is carried out using a node-centred approach. If mapping fails, the return monitoring technique is used to replenish previous activities with the same II to avoid declines in results.

GraphMinorformalizes the mapping problem so that a short, isomorphic MRRG subparagraph in DFG diagrams can be identified. GraphMinor provides a path-sharing technology to preserve knowledge from a manufacturer over a lifetime for many PE users. This process improves the use of PE substantially. But GraphMinor does not optimise the use of CGRA buffer resources.

In summary, DRESC, PSOMap and AA-ILP constitute the relatively widespread CGRA mapping resolution for all DFG formats. In theory, the optimal solution will converge in time. However, this time the tuning parameters are unknown in the algorithm. In the sense of time compilation, the tunable parameters must be changed for

sacrificing the output. The compilation time may otherwise be unacceptable, in particular when compiling broad and irregular loops.

For heuristics such as EMS, resource-aware and GraphMinor, advanced optimization heuristics have trouble with mapping and include one or more basic optimizations that help you find the optimal solution quickly. They are easily compared to maps with classic heuristic methods, but loose efficiency and generality.

4. Decomposed Mapping

SPR disrupts preparation, positioning and distribution. SPR requires transactions to be repositioned outside of slack windows in order to accommodate data latency movements, but its mechanism is nearly as dominant as DRESC.

The additional function of EPIMap recomputations explains the problem of graph epimorphism. A systematic approach is used to programme every operation (performance schedule) and transform the DFG into an amended map to meet various restrictions. The positioning and routing method for the time-extended CGRA (TEC) graph is converted into the updated DFG as much as possible. The mapper produces a compatible diagram between DFG and TEC, which describes each node as a possible mapping pair, to detect a subgraph. Due to mapping restrictions, certain edges are missing. Finally, for a compiler with nodes that match the number of updated DFG operations, a maximum click is required.

For longer life, EPIMap prefers PE as a machine-based routing engine. REGIMap is employed by the LRB to buffer the value. Combines buffers with locations and routing, so that substantial overhead restoration due to register shock can efficiently be avoided. Memory-conscious is a directly used OMB plotting algorithm. OMB is not the last choice for a memo-mapping method to deal with registration dumping. RAMP is an improvement in REGIMap, allowing heuristic re-planning to correct mapping errors in order to avoid the lengthy search time for the compiler.

A robust routing solution is given by RAMP. In terms of output and time compared to other mappings, it is now highly competitive. The discovery of the maximum click in EPIMap, REGImap and RAMP is however an NP-complete problem as the chart is compatible with the broad chart with the thickness of relation and the time complexity of paper heuristic usage is O (N8). The main problem is that the map does not specify if the whole click is open. Although RAMP offers strong programming heuristics, the problem is still present.

The key thing is a precise space and time mapping mechanism. After time mapping, the impossible space mapping by DFG should be expected, and the compiler should realize this impossibility without too much time. Our mapping uses therefore a completely different concept of time and space mapping. In order to ensure spatial mapping accuracy, we believe that time mapping can effectively optimise and thoroughly evaluate DFG.

The targets should be achieved: The time of each DFG operation must be determined in accordance with the available routing resource within CGRA and routing solutions for all routing paths within the DFG should be offered. For optimization purposes all buffer sources like PE, LRB, GRB and OMB should have been used. Effective algorithms for the routing resolution of CGRA interconnections should be used to verify that space maps are pinged over DFG after time mapping.

To ensure that the CGRA PEs are enough to support the parallelism generated during temporal mapping and reschedule time for certain operations, a strong calculation limiting algorithm should be utilised.However, instead of time consumption, spatial algorithms should be quick and efficient.

5. Temporal Mapping

The temporary mapping method receives the initial DFG required for spatial mapping as the entry and output of an amended DFG. All its nodes with a fixed time schedule and their data are assigned to a buffer resource are part of the updated DFG. Additional LRB restrictions are added when data is buffered into LRB as shown in figure 1. The GRB buffering and consumption of generated data in operation A in E, F, and G are defined by the routing solution. B data is tamped and consumed in LRB by F. The same PE needs the B and F mapping.

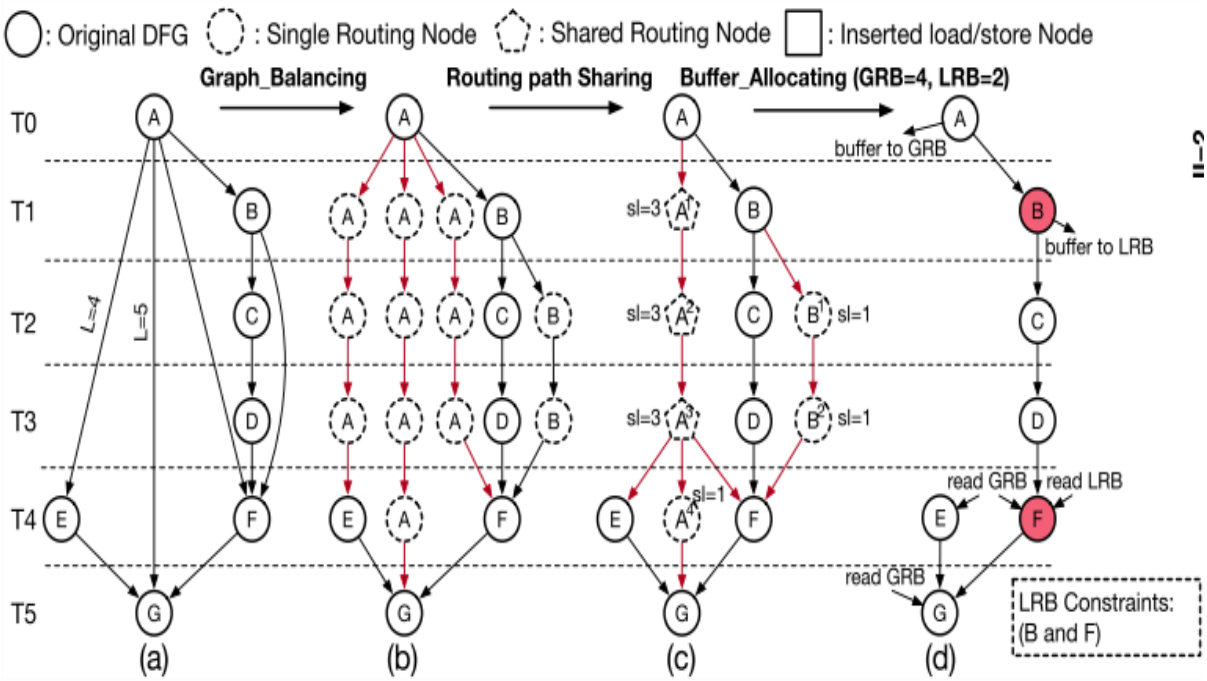

Figure 1: Data Flow Diagram (DFG)

Temporal flow is shown in figure 2 .Twomain algorithms are included in our temporary mapping phase, including time allocation, graph balance, RPS sharing, buffer allocation, interconnection constraints solving (ICS) and computational constraints solving (CCS). Iteratively the algorithms flowed until the DFG felt it is ready for space mapping. The three algorithms are briefly implemented, mainly by means of existing and the latter.

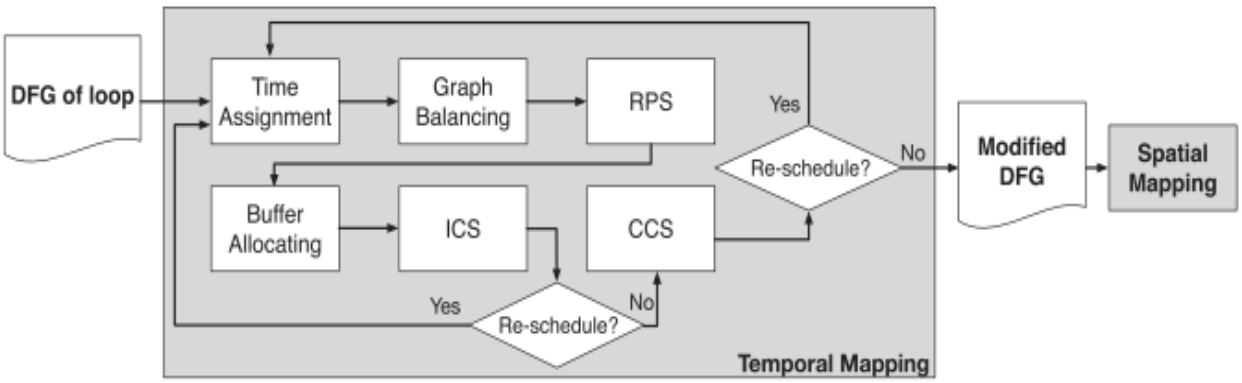

Figure 2: Temporal flow

5.1 Buffer Allocating Algorithm

Input: $D_m(V_m, E_m)$, II and BI
Output: $D_b(V_b, E_b)$, SPT
$FinishAllocation \leftarrow Failed$;
$GRPs, LRPs, ORPs \leftarrow \emptyset$;
while $FinishAllocation = Failed$ **do**
  $D_b \leftarrow D_m$;
  $RPs \leftarrow GenerateRoutingPaths(D_b)$;
  for *each* $P_i \in RPs$ *of* $D_b, 0 \leq i < |RPs|$ **do**
    if $LRB\_constraints\_checking(S_p^i, II, BI) = True$ **then**
      $LRPs \leftarrow LRPs \cup P_i$;
      $RPs \leftarrow RemovePath(RPs)$;
    end
  end
  $GRPs, LRPs, RPs \leftarrow AssignGRB(RPs, BI, GRPs, LRPs)$;
  for *each* $P_i \in RPs$ *of* $D_b, 0 \leq i < |RPs|$ **do**
    $LRPs, ORPs \leftarrow CombAssign(P_i, II, BI)$;
  end
  $LRPs \leftarrow MergeLRBPaths(D_b)$;
  if $\lceil |V_b|/N_{pe} \rceil > II$ **then**
    $II \leftarrow II + 1$;
    continue;
  else
    $D_b \leftarrow GRBAssignRule(GRPs)$;
    $D_b, SPT \leftarrow LRBAssignRule(LRPs)$;
    $D_b \leftarrow OMBAssignRule(ORPs)$;
    $FinishAllocation = Success$;
  end
end

5.2 GRB Assignment Algorithm

Input: $RPs, BI, GRPs, LRPs$
Output: $GRPs, LRPs$
$RPs \leftarrow SortRoutingPaths(RPs)$;
$GRPs \leftarrow InitializeGRPs(RPs)$;
$max\_sl \leftarrow getMaxSharedLevel(RPs)$;
for $max\_sl \geq$ **shared_level** > 0 **do**
  for *each routing path* $P_i(V_p^i, E_p^i)$ *in ordered* RPs **do**
    for *each* u_j^i *in* V_p^i, *j from 1 to* J^i **do**
      if $u_j^i.sl =$ ***shared_level*** **then**
        $G_i \leftarrow extendPath(G_i, u_j^i)$;
        $L_i \leftarrow OCRP(G_i, P_i)$;
        if $GRB\_constraints(GRPs, II) = False$ **then**
          $G_i \leftarrow remove(G_i, u_j^i)$;
          $RPs \leftarrow UpdateRPs(GRPs)$;
          return $GRPs, RPs$;
        end
        if $LRB\_constraint\_checking(S_l^i) = True$ **then**
          $LRPs \leftarrow LRPs \cup L_i$;
          $RPs \leftarrow RemovePath(P_i)$;
        end
      end
    end
  end
end
$RPs \leftarrow updateRPs(GRPs)$;
return $GRPs, RPs$;

6.Results and Discussions

We construct a complete LLVM stack compiler system to check the efficiency and time of our mapping algorithm. The compiler extracts and compiles the loops in CGRA from various applications. To test the performance of the compiler, a runtime simulator is developed. The base line, REGIMap and RAMP are the two advanced algorithms of mapping we choose. These two algorithms for mapping constitute two representative mapping algorithms that describe space and time and are competitive in efficiency and composition time with other CGRA MS algorithms.The language C loop for the true dynamic energy-efficiency reset fabric targeting the Berkeley 13 diagrams can be extended through our compiled fluctuations.

You can modify the Clang by inserting a keyword by selecting the kernel on the loop (LLVM front-end). The loop kernel is extracted from the principal LLVM IR programme and designed the DFG from the LLVM IR. Our mapping algorithm is constructed and applied while the LLVM backend goes through it. Our mapping algorithm converts the REGIMap and the RAMP open-source code (htts:/github.com/cmlasu/ccf) to the LLVM frame to allow a reasonable comparison between DFG outputs and build times. 4x4 PEs with various GRB and LRB sizes have been experimented by CGRAs. Since PE is connected to a 2D torus network, all PEs can enter information and travel from neighbouring PE. Often on column or line are the first PE or last PE. Both tests are performed with a 3.60 GHz CPU frequency on the same Intel Core i5 computer.We take 28 computer-intensive loop kernels from different applications and suites. Current benchmarks, such as MachSuit and Polybench, are used to derive benchmarks such as EEM bench and MediaBank and MiBench. Also revised are digital signals processing (DSP), graphic search and dynamic programming (DP) and computer vision from various application areas. The operating scale ranges between 17 and 154. A regularity of the data flow charts is determined by the number of routing paths and high-quality nodes within the DFG. The RPs indicate that the relationship between the producer and the client is higher than 1. More RPs can be found. DFG means improved compilation pressure buffer control. A node edge and an input edge are at each operation stage. Those with three inputs (same instructions) or multiple outputs are generally very classic (larger than 1). With the highest quality nodes in the DFG, the compiler manages more linked resources.

Table 2: Average performance of complied loops compared with REGIMap

Average performance of complied loops(MII/II)	LRB=2,GRB=0		LRB=2,GRB=4		LRB=4,GRB=4		LRB=8,GRB=4	
	REGIMap	Our Method	REGIMap	Our Method	REGIMap	Our Method	REGIMap	Our Method
	0.82	0.93	0.82	0.94	0.87	0.95	0.87	0.96

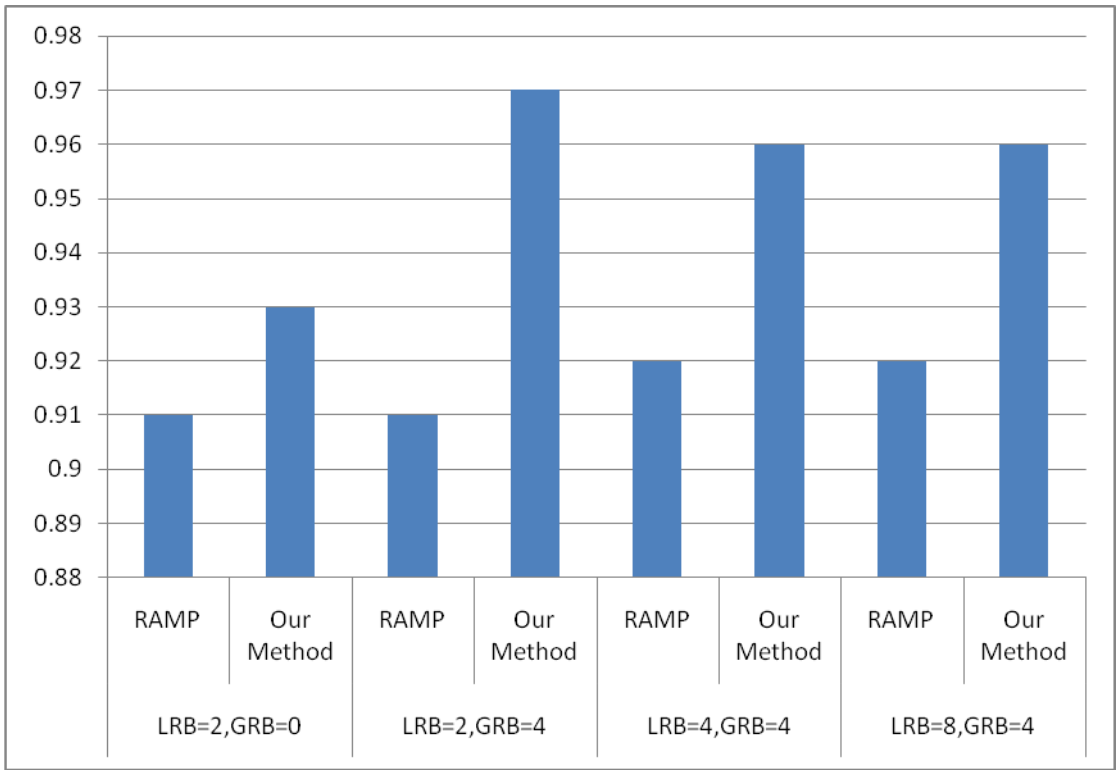

Figure 3: Average performance of complied loopscompared with REGIMap

Table 3: Average performance of complied loopscompared with RAMP

Average performance of complied loops(MII/II)	LRB=2,GRB=0		LRB=2,GRB=4		LRB=4,GRB=4		LRB=8,GRB=4	
	RAMP	Our Method	RAMP	Our Method	RAMP	Our Method	RAMP	Our Method
	0.91	0.93	0.91	0.97	0.92	0.96	0.92	0.96

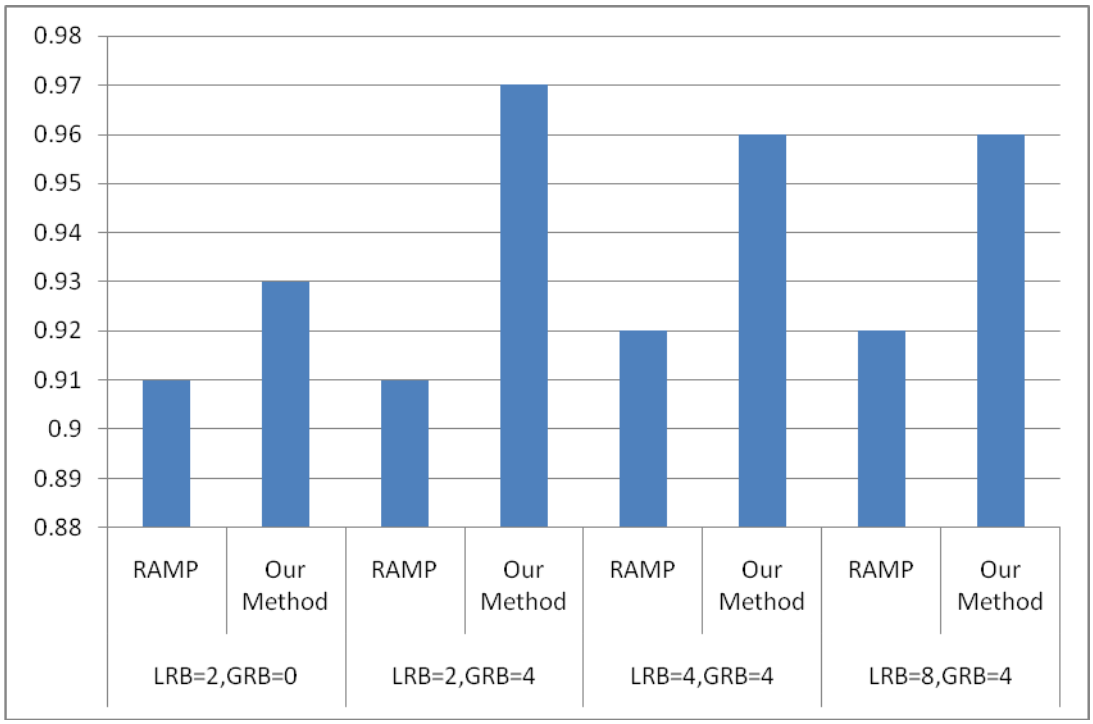

Figure 4: Average performance of complied loopscompared with RAMP

Table 4: Comparison of Compilation successful rate

	LRB=2,GRB=0		LRB=2,GRB=4		LRB=4,GRB=4		LRB=8,GRB=4	
Compilation successful rate	REGIMap	89.1	REGIMap	89.1	REGIMap	89.1	REGIMap	89.1
	RAMP	85.5	RAMP	85.5	RAMP	92	RAMP	92
	Our Method	99	Our Method	99	Our Method	99	Our Method	99

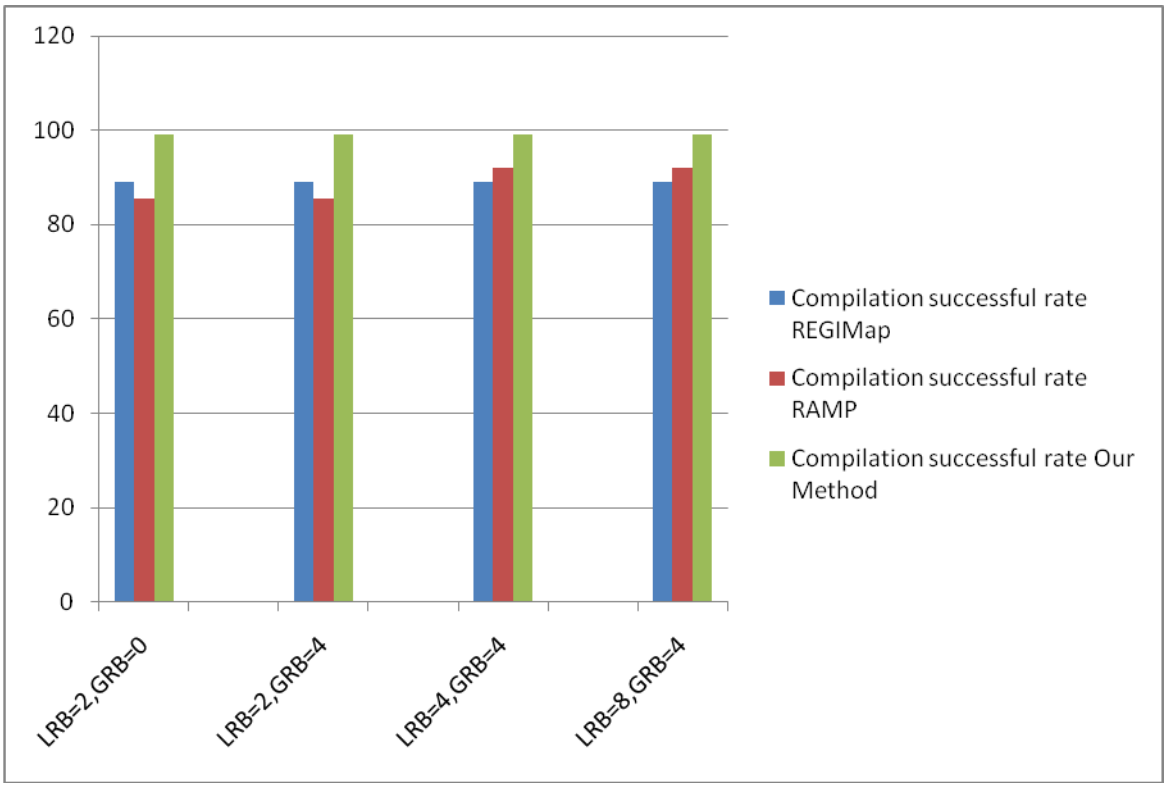

Figure 5: Comparison of Compilation successful rate

Table 5: Compilation Time compared with RAMP

	LRB=2,GRB=0		LRB=2,GRB=4		LRB=4,GRB=4		LRB=8,GRB=4	
Compilation Time(Sec)	RAMP	Our Method	RAMP	Our Method	RAMP	Our Method	RAMP	Our Method
	55.4	15	55.3	5	95	8	81	8.2

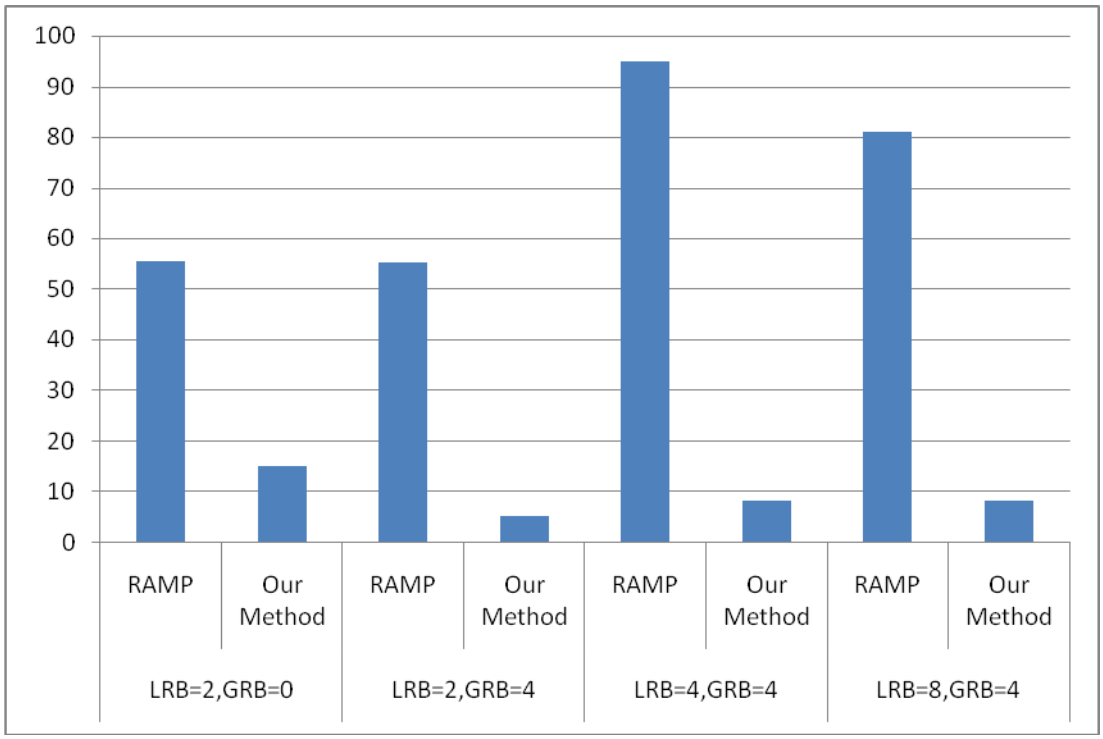

Figure 6: Compilation Time compared with RAMP

Table 6: Compilation Time compared with REGMap

	LRB=2,GRB=0		LRB=2,GRB=4		LRB=4,GRB=4		LRB=8,GRB=4	
Compilation Time(Sec)	REGMap	Our Method	REGMap	Our Method	REGMap	Our Method	REGMap	Our Method
	30	11	30.5	6	23	9	22	9

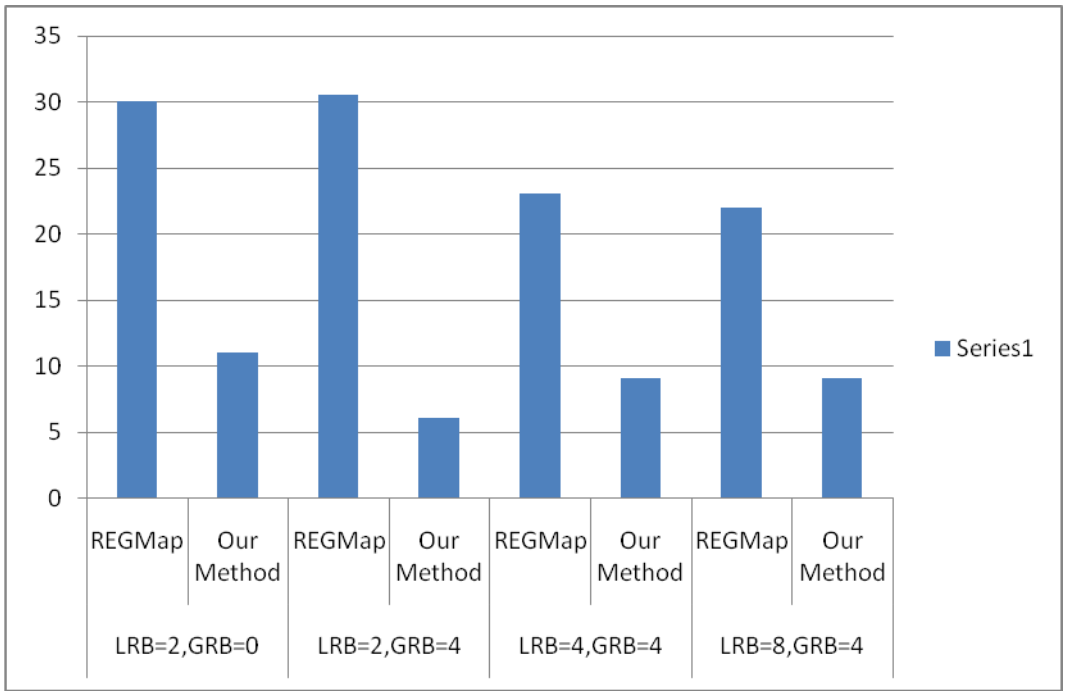

Figure 7: Compilation Time compared with RAMP

6.1 LRB Size influence

Fig. 3-7 demonstrates REGIMap efficiency, RAMP efficiency and our approach to mapping. The Y axis is the minimum ratio of II and actual II, which indicates the best theoretically.

6.2 REGIMap Vs Our Method

Table 2 shows that if LRB=2, REGIMap is an average of 0.82standard loop efficiency and 0.93 for GRB=0 and 0.96 for GRB=4 for our mapping. If LRBs are between 2 and 4 and 8, the REGIMap average output is between 0.82 and 0.87 and up to 0.95 and 0.96. With LRB growing by 2 to 4 and 8 REGIMap raises their average output by 0.82 to 0.87.

6.3 RAMP Vs Our Method

Table 3 shows that four LRB=2 loop kernels are unable to be mapped by RAMP. It shows that the RAMP is suitable for successfully mapping 0.93 loops.The result is 0.93 for GRB= 0 and 0.96for GRB= 4 for our mapping algorithm. RAMP may have another two mapped and unstructured loops with increasing LRB sizes. In all mapped kernels, the average output improves slightly. RAMP reveals more routing than REGIMap and RAMP is better than REGIMap than decent kernels. This is critical for improving RAMP.

REGIMap provides more programming options when the map fails, and OMB is used during the time mapping procedure explicitly. However, RAMP has selected only one technique to devote buffer resources to this.

The hybrid use of these routing options, particularly the productive use of the major GRB, is not taken into consideration.

6.4 GRB Size Influence

Fig. 8 indicates an average of PE and GRB mapping time (GRB=0, 4, 8, 16) and 4x4 CGRA mapping for both PE and GRB scaled mapping. We can observe a relatively improved performance with increasing GRB scale. This data shows that our mapping is highly adaptive as 95% of its theoretical best results are not contained in CGRA although it does not contain any GRBs. However, the lack of GRB will increase the time of compilation. This is because LRB and OMB have more routes. The assignment of the LRB routing path has the same PE mapping limit. This mapping limitation would increase the likelihood of a potential mapping failure and would lead to space monitoring and rearrangement to prevent a decrease in efficiency.

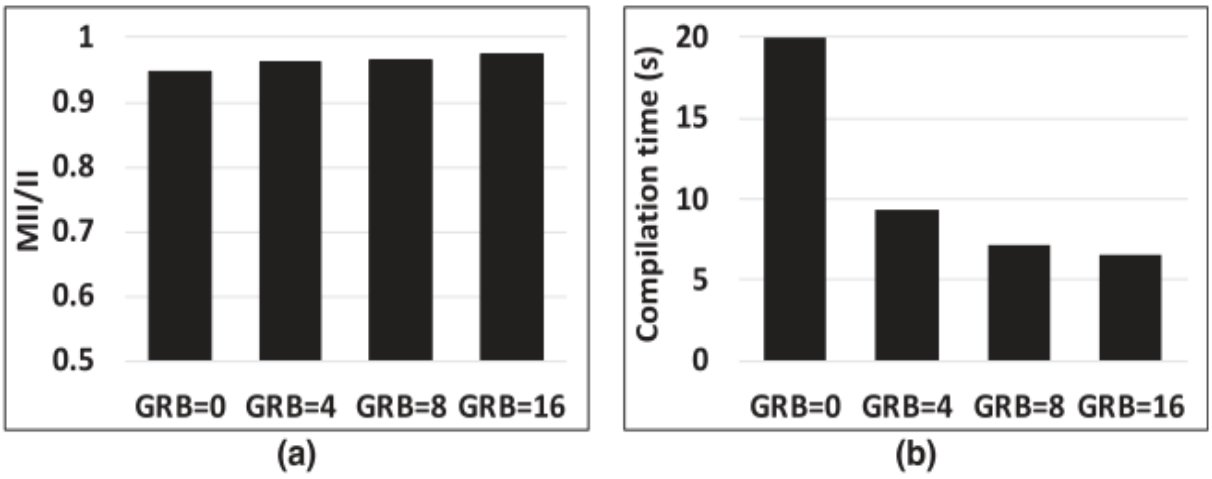

Figure 8: (a) Performance of our mapping over 4X4 CGRA with LRB=2 ateach PE and GRB=0, 4, 8 and 16, (b) compilation time of our mappingover 4x4 CGRA with LRB=2 and GRB=0, 4, 8, 16.

7. Conclusion

In this paper, we explore with a new approach the topic of modular CGRA planning. We've used the temporal mapping policy to argue that the time mapping flow is considerably higher than with decomposed maps. The structural development concentrates on a temporary flow of mapping which includes a large buffer for heuristic and connecting algorithms and computer resources, restricts cartoon output and ensures the below spatial mapping success rate. Our approach can produce a large rate of success when mapping in some time budgets, in combination with lightweight, fast and powerful spatial mapping algorithms. In addition, our experimental results show that our mapping can produce more power and less compilation time.

REFERENCES

1. C. Kim, M. Chung, Y. Cho, M. Konijnenburg, S. Ryu, and J. Kim, “ULP-SRP: Ultra low power samsung reconfigurable processor for biomedical applications,” in Proc. Int. Conf. Field-Programmable Technol., 2012, pp. 329–334.
2. D. Shukla and A. Johari, "Study, Design and Analysis of 8 bit MIPS Processor using deepsubmicron CMOS C5 process," 2018 International Conference on Advanced Computation and Telecommunication (ICACAT), Bhopal, India, 2018, pp. 1-5,
3. S. An et al., "C-MAP: Improving the Effectiveness of Mapping Method for CGRA by Reducing NoC Congestion," 2019 IEEE 21st International Conference on High Performance Computing and Communications; China, 2019, pp. 321-328
4. F. Bouwens, M. Berekovic, A. Kanstein, and G. Gaydadjiev, “Architectural exploration of the ADRES coarse-grained reconfigurable array,” in Reconfigurable Computing: Architectures, Tools andApplications. Berlin, Germany: Springer, 2007, pp. 1–13
5. N. N. Qaqos, "Optimized FPGA Implementation of the CRC Using Parallel Pipelining Architecture," 2019 International Conference on Advanced Science and Engineering (ICOASE), Zakho - Duhok, Iraq, 2019, pp. 46-51
6. S. A. Chin et al., “CGRA-ME: A unified framework for CGRA modelling and exploration,” in Proc. Int. Conf. Appl.-Specific Syst. Archit.s Processors, 2017, pp. 184–189.
7. M. Bachir, A. Cohen and S. Touati, "On the effectiveness of register moves to minimise post-pass unrolling in software pipelined loops," 2012 International Conference on High Performance Computing & Simulation (HPCS), Madrid, Spain, 2012, pp. 551-558
8. B. De Sutter, P. Coene, T. Vander Aa, and B. Mei, “Placement-and- routing-based register allocation for coarse-grained reconfigura- ble arrays,” in Proc. ACM SIGPLAN-SIGBED Conf. Lang. Compilers, Tools Embedded Syst., 2008, pp. 151–160.
9. M. Hamzeh, Compiler and Architecture Design for Coarse-GrainedProgrammable Accelerators. Phoenix, AZ, USA: Arizona State Uni- versity, 2015.
10. M. Hamzeh, A. Shrivastava and S. Vrudhula, "REGIMap: Register-aware application mapping on Coarse-Grained Reconfigurable Architectures (CGRAs)," 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 2013, pp. 1-10
11. S. Yin et al., “Conflict-free loop mapping for coarse-grained reconfigurable architecture with multi-bank memory,” IEEE Trans. Par- allelDistrib. Syst., vol. 28, no. 9, pp. 2471–2485, 2017.

12. S. Dave, M. Balasubramanian, and A. Shrivastava, "RAMP: Resource-aware mapping for CGRAs," in Proc. 55th Annu. Des.Autom. Conf., 2018, pp. 127:1–127:6.
13. M. Balasubramanian and A. Shrivastava, "CRIMSON: Compute-Intensive Loop Acceleration by Randomized Iterative Modulo Scheduling and Optimized Mapping on CGRAs," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 11, pp. 3300-3310
14. Q. Liping, M. Yan, L. Dongheng and X. Hai-Bo, "A Quantum Particle Swarm Optimization Algorithm with Available Transfer Capability," 2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), Xuzhou, China, 2020
15. H. Park, K. Fan, S. A. Mahlke, T. Oh, H. Kim, and H.-S. Kim, "Edge-centric modulo scheduling for coarse-grained reconfigurable architectures," in Proc. 17th Int. Conf. Parallel Archits Compilation Techn., 2008, pp. 166–176.
16. J. D. Souza, L. Carro, M. B. Rutzig, and A. C. S. Beck, "A reconfigurable heterogeneous multicore with a homogeneous ISA," in Proc. Des. Autom. Test Europe Conf. Exhibit., 2016, pp. 1598–1603.
17. H. Mahdi, A. Shrivastava, and S. Vrudhula, "EPIMap: Using epi- morphism to map applications on CGRAs," in Proc. Des. Autom. Conf., 2012, pp. 1280–1287.
18. M. Pham, D. B. Hoang and Z. Chaczko, "Congestion-Aware and Energy-Aware Virtual Network Embedding," in IEEE/ACM Transactions on Networking, vol. 28, no. 1, pp. 210-223, Feb. 2020.
19. S. A. Chin and J. H. Anderson, "An architecture-agnostic integer linear programming approach to CGRA mapping," in Proc. 55th ACM/ESDA/IEEE Des. Autom. Conf., 2018, pp. 1–6.
20. T. Nowatzki, M. Sartin-Tarm, L. De Carli, K. Sankaralingam, C.Estan, and B. Robatmili, "A general constraint-centric scheduling framework for spatial architectures," in Proc. 34th ACM SIGPLANConf. Program. Lang. Des. Implementation, 2013, pp. 495–506.
21. ADILMOUSSEBBIH, D., SOUISSI MOHAMED, D., ABDELKADER, L., & MOHAMED, F. MODELING AND MAPPING OF THE WATER EROSION RISK USING GIS/RUSLE APPROACH IN THE BOUREGREG RIVER WATERSHED.
22. Slimani, R., & Guendouz, A. (2015). Groundwater vulnerability and risk mapping for the Phreatic aquifer in the Ouargla Oasis of Algerian Sahara using GIS and GOD method. International Journal of Agricultural Science and Research (IJASR), 1(5), 149-158.
23. Abed, S. S., & Abbas, R. F. (2016). S-iteration for general quasi multi valued contraction mappings. Int. J. Appl. Math. Stat. Sci, 5 (4), 9, 22.
24. Indu, M., & Somasundareswari, D. (2015). Energy Efficient Carry Skip Adder Using Skip Logic In Various Voltage Levels. International Journal of Management, Information Technology and Engineering (BEST: IJMITE), 3(10), 145-150.
25. GUPTA, V., & SAXENA, G. COMMON FIXED POINT THEOREM FOR FINITE NUMBER OF WEAKLY COMPATIBLE MAPPINGS IN QUASI-GAUGE SPACE.