Parallel Processing of Sequential Media Algorithms on Heterogeneous Multi-Processor System-on-Chip

Peng Zhao
School of Computer Science, National University of Defense Technology, Changsha, China
Email: pengzhao@nudt.edu.cn

Dawei Wang, Ming Yan and Sikun Li
School of Computer Science, National University of Defense Technology, Changsha, China
Email: {wangdawei@nudt.edu.cn, minyan@nudt.edu.cn, lisikun@263.net.cn}

***Abstract*—Heterogeneous Multi-Processor System-on-Chip (MPSoC) and media processing are comprehensively applied in mobile electronic commerce. And heterogeneous MPSoCs provides more opportunities for parallelization accelerating of sequential media algorithms. However, the parallelization researches of heterogeneous MPSoC applications lags far behind the development of MPSoC hardware platform. Therefore, utilizing parallelization opportunity of MPSoC for improving performance and efficiency of media applications has been one of the hottest researches in the field of embedded system. This paper proposes a new approach that parallelizes sequential media algorithms on heterogeneous MPSoC using program transformation and application-to-architecture mapping techniques. Data locality and communication cost are optimized during the parallel processing. Moreover, the difference between processing elements, reflected in architecture templates, is used to achieve "the maximum" performance and efficiency of heterogeneous MPSoCs. Finally, an experiment shows the proposed approach can obtain approximate or better accelerating than the manual parallel processing by experienced designers.**

***Index Terms*—heterogeneous Multi-Processor System-on-Chip, media processing, parallel processing, application mapping**

I. INTRODUCTION

In electronic commerce business, potential buyers acquire product information almost entirely from vision including text, image, video and digital geometry, the latter three of which are called by a joint name: visual media. Visual media, as an instrument to show the charm of your commodity, greatly determines whether the commodities are welcomed and what degree they are welcomed. Visual media plays a very important role in the electronic commerce. Visual media has the following features mainly including frequent loop nests, enormous multi-dimension signals processing, lacking of determinate structures, multi-dimension signals, and semantic diversity [1]. Visual media processing can be regarded as a mixture of processing image, video and digital geometry. This paper just focuses on the embedded visual media processing in mobile electronic commerce.

Multi-Processor System-on-Chip (MPSoC) is comprehensively applied in the mobile electronic commerce. Recent years have proved that MPSoC is the most promising and feasible way to achieve high integration provided by the semiconductor technology under the market constraints of performance and power consumption [4]. Due to the high computational and power consumption demands of modern embedded applications, especially for embedded visual media processing, MPSoC architectures often contain multiple heterogeneous processing elements. From some view points, heterogeneous MPSoC lays the physical groundwork for mobile electronic commerce, but the difference of processing elements makes the parallel processing of sequential algorithms more challenging. Parallel processing of heterogeneous MPSoC has been a critical problem and one of the hottest research points in embedded system.

In order to address the problem, some new programming models, such as OpenMP [2] and MPI [3], have been proposed. Usually, such solutions require the programmer to heavily modify the original source code manually. And the ideal desirability is to hide the complexity of parallel programming from the programmers as far as possible [5].

Other new models of computation (MoC) or languages, like task graphs [6] and Kahn Process Networks (KPNs) [7], have also been proposed. These MoCs and languages require a complete rewrite of the application with a new language which most embedded programmer may not be familiar with [9]. Because the huge amount of legacy C-language codes are often used as MPSoC software, no new parallel programming models or languages have been widely adopted in MPSoCs so far. Therefore, sequential C implementations often need to be parallelized. Unfortunately, efficient

compiler support for this purpose is not available until today [4].

One of parallel processing methods of sequential media algorithms on heterogeneous MPSoC is application-to-architecture mapping. The application mapping means the mapping of functionality into computation and storage resources, which can also achieve parallelism of multiple processing elements. And several such tools have been developed. The Sesame framework [10] supports design space exploration of heterogeneous embedded multimedia systems and provides useful tools. Metropolis [12] provides users of interfaces for specifying functional models, architecture models and mapping models. It also supports the simulation of these models.

However, in these tools the application-to-architecture mapping is performed manually by experienced designers. With increasing designs complexity, the application mapping is becoming very difficult and error-prone. Therefore, automatic application mapping has been one of hottest researches and some researchers have studied this key problem [13, 14, 15]. The automatic mapping problem in Multi-Processor systems is NP-complete and traditionally addressed as heuristics or Integer Linear Programming (ILP) problems [13, 18, 19, 20]. But these automatic mapping methods do not address coarse-grained tasks mapping to processing elements or are not fully aware of the difference between the processing elements of heterogeneous MPSoC, which will be further discussed in Section 2.

This paper proposes a new approach that parallelizes sequential media algorithms on heterogeneous MPSoC using program transformation and application-to-architecture mapping techniques. The proposed approach focuses on the parallel processing of critical loops. It tests dependence relation between multi-dimension signals and extracts coarse-grained parallelism from sequential algorithms using static program analysis and tiling transformation, and then parallelizable mapping candidates, tiles, with optimal data locality and communication cost are generated. Using eACOGA algorithm, the tiles are mapping to the architecture template that is modeled to reflect specific architecture including differences between processor cores of heterogeneous MPSoC in an abstract way.

The rest of this paper is organized as follows. In section 2, a summary of the related works is presented. Section 3 formulates the parallel processing problem. Section 4 describes the VMP-MPSoC (Visual Media Processing – Multi-Processor System-on-Chip) [16] architecture template. Section 5 presents the parallel processing methodology. Section 6 gives the experimental results and analysis. Finally, section 7 concludes this work and gives insights into the future work.

II. RELATED WORKS

In this paper, sequential media is parallel processed on bus-based heterogeneous MPSoCs mainly by static application-to-architecture mapping. The recent researches on static application-to-architecture mapping mainly address Integer Linear Programming (ILP) or heuristic solutions.

Martino Ruggiero [13, 17] implements an efficient and exact approach to the mapping problem based on a decomposition strategy. The allocation sub-problem is solved through IP. Its target architecture is a general template for homogeneous multi-core architectures.

Kugan Vivekanandarajah [18] proposed an algorithm using efficient branch-and-bound approach to partition the mapping problem into sub-problems and solves them. Its target architecture is general heterogeneous MPSoC architectures that defined with architecture graphs.

Pierre-Andre Mudry [14] partitioned SoC application into hardware and software parts using mixture genetic algorithms with system constraints. Their proposed method is in the context of the Move architecture, which belongs to the class of Transport Triggered Architectures (TTA).

Cagkan Erbas 2006 [15] consider the application mapping of heterogeneous embedded systems as multi-objective (processing time, power consumption and architecture cost) optimization problem.

Application mapping onto heterogeneous MPSoCs can be regarded as one way of program parallelization. And parallelism is a key factor in the parallel processing and application-to-architecture mapping. However, the above approaches only address the mapping problem but exclude the coarse-grained parallelism problem of heterogeneous MPSoC applications, which results in the mapping schemes possibly with low parallelism. Moreover, the above approaches are not fully aware of the difference between the processing elements of heterogeneous MPSoCs. Heterogeneous MPSoCs challenge traditional mapping approaches for homogeneous MPSoCs. The difference between processing elements should be considered in the parallel processing, less that “the optimal” parallel processing scheme cannot be achieved.

III. ROBLEM FORMULATION

Mobile electronic commerce requires less execution time and power consumption when processing visual media applications, which challenges the parallel processing of sequential media algorithms on heterogeneous MPSoC. Here, the proposed approach parallelizes sequential media algorithms by application-to-architecture mapping onto heterogeneous MPSoC. And the problem is formulated here.

A. Application Model

Program Information Aided Control-Dataflow Task Graph (PIA-CDTG) [21] is used to model applications. PIA-CDTG model attains Task Graph with program features such as control flow/data flow, computation/storage distribution, and critical loops features. PIA-CDTG model can be extracted using program analysis techniques [21].

Definition 1. (PIA-CDTG model) PIA-CDTG model is defined as $PIA-CDTG=\{N,E,P\}$ where

- *N* is a node set. Each node is a task.
- $E = \{n_i > n_j \mid n_i, n_j \in N\}$ is a set of directed edges. $n_i > n_j$ represents a directed edge from n_i to n_j. Directed edges can be divided into two types: control flow edges and data flow edges.
- *P* defines the propriety of PIA-CDTG to represent the features of application programs.

Media processing program execution tends to spend most of the time in frequent loop nests. By their very nature, MPSoC applications tend to follow the rule even more so than desktop type of application [22]. In this paper, the proposed approach focuses critical loops, those loop nests which appear frequently and consume much time every execution. Critical loops are represented with vi-DFG models, which are defined as follows.

Definition 2. (vi-DFG model) vi-DFG model is defined as vi-DFG = (*N*, *E*) where

- *N* is a node set. Each node describes the instance of the 8 virtual instructions defined by Kahn process.
- $E \subseteq (T \times T)$ is a set of directed edges. Each edge describes the interconnection (such as loop start, loop end, data dependency, data copy and write control) between virtual instruction nodes.

In the parallel processing of sequential media algorithms, mapping candidates are generated on the basis of polyhedral-model-based data dependency analysis. The related definitions of polyhedral model are given as follows.

Definition 3. (Affine hyperplane) All the vectors $x \in Z^n$ such that $\mathrm{h}.x = k$ compose of an affine hyperplane where $k \in Z$.

Definition 4. (Polyhedral model) All the vectors $x \in Z^n$ such that $Ax + b \geq 0$ define a polyhedral model with integer vertex where A is an integer matrix. A polytope is a bounded polyhedron.

Mapping candidate generation is based on data dependency. In this paper, dependency is described using enhanced data dependency graph, which is defined as follows.

Definition 5. (Enhanced data dependency graph) The enhanced data dependency graph can be described as directed multi-graph $G_i =< S, T, \mathrm{P}_e >$, for $T \subseteq R$, $R = S \times S$ and R being transitive closure of T. Each vertex in G_i is one statement. Each edge $e_{ij} \in T$ from vertex S_i to vertex S_j represents dependency from execution instance S_i to execution instance S_j. Supposed $(S_i, S_j) \in R$, S_j should be executed before S_i.

The enhanced data dependency graph extends the data dependency graph by adding one additive property: dependency polyhedral model P_e, which describes all the data dependency and greatly helps the data reuse analysis. The dependency polyhedral model P_e is defined as follows. The equality in P_e describe an affine relation from destination iteration vector t to source iteration vector s that accesses the conflict memory source last time.

$$\mathrm{P}_e = \begin{bmatrix} D^{\mathrm{P}_{\vec{s}}} \\ \hline D^{\mathrm{P}_{\vec{i}}} \\ h_e \end{bmatrix} \begin{bmatrix} s \\ t \\ p \\ 1 \end{bmatrix} \begin{bmatrix} \geq 0 \\ \hline = 0 \end{bmatrix}$$

B. Architecture Model

The parallel processing of sequential media algorithms onto heterogeneous MPSoC needs an abstract representation of MPSoC architecture model. The architecture model is represented as an architecture graph, which is defined as follows.

Definition 6. (Architecture graph) The architecture graph $G_A = (V_A, E_A)$ is used to model MPSoC architectures. V_A denotes the set of all the architecture components. Let $v \in V_A$, and then $type(v) \in \{pe, mem, bus, io, other\}$. E_A denotes the connection between architecture components. The edge $e \in E_A$ represents the communication medium and will have a bandwidth capability associated with it.

C. Parallel Mapping Problem

In this problem model, *n* parallel tiles have to be allocated to *m* heterogeneous processing elements. That is, there are *m* processing elements and *n* tiles after the generation of mapping candidates.

Definition 7. (Tile-to-architecture mapping problem) Given a tile set $\Gamma : \{T_1, T_2, ..., T_n\}$ and an architecture graph $G_A = (V_A, E_A)$, each tile in Γ carries three kinds of representation: vi-DFG, source code and SUIF [25] code. Each tile has an instruction matching value $\mathrm{I}_{i,j}$ and a operand matching value $\mathrm{D}_{i,j}$ where $1 \leq i \leq n$ and $1 \leq j \leq m$. The matching values for the tile T_i depends on which of the PE it goes to and denoted by subscript *j*. Mapping tiles onto MPSoC can be regarded as finding a matrix $\bar{x} = \{x_{i,j}\}$ such that the matching function is maximized, where $1 \leq i \leq n$ and $1 \leq j \leq m$. $x_{i,j} \in \{0, 1\}$ denotes one if tile T_i is mapped to PE *j* and zero otherwise.

The overall matching value is defined as follows:

$$MF = \max\{\alpha \cdot I_{i,j} + \beta \cdot D_{i,j}\} \text{ where}$$

$I_{i,j}$ denotes the matching value between instruction sequence features of tile T_i and instruction set features of PE *j*. $D_{i,j}$ denotes the matching value between operand features of tile *i* and PE *j*. $0 \leq \alpha, \beta \leq 1$ are the tuning coefficients.

The parallel mapping should subject to the following constraints:

- One tile is mapped only to one PE,

$$\sum_{i=1}^{n} x_{i,j} = 1 \qquad for \quad j = 1,2.....n$$

- All the tiles are assigned PE,

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j} = m$$

This is an NP-Complete resource allocation problem [26]. Solving on mapping optimization problems using Ant Colony Optimization algorithms (ACO) achieves good effects. But the strategy of random selection in constructing solutions leads to slow convergence speed. Furthermore, the principle of positive feedback can not only strengthen the solutions with better performance, but also bring on the stagnancy of search. To address this problem, an eACOGA algorithm [28, 29] is proposed, which will be presented in section 5.

Ⅳ. TARGET ARCHITECTURE

A. MPSoC Architecture

Heterogeneous MPSoC is comprehensively applied in the visual media processing field. And VMP-MPSoC [16] is exactly this kind of products. Processing visual media that comprises of image, video and digital geometry puts forward a special claim for MPSoC performance and efficiency, which evokes this innovative VMP-MPSoC architecture.

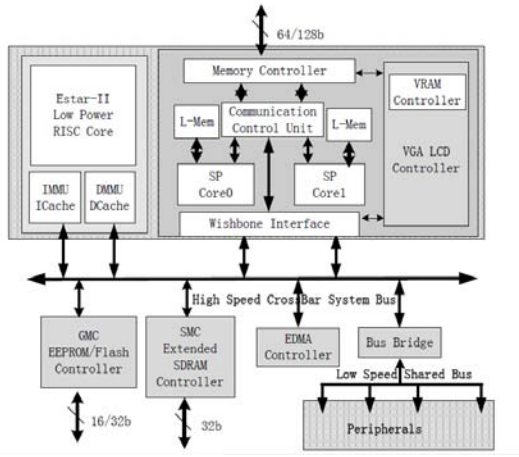

Figure 1. VMP-MPSoC architecture

VMP-MPSoC takes EStarII [27], an autonomous designed 32-bit Embedded RISC processor, as the host processor, and two synergistic SIMD processors [16] with intellectual property protection to accelerate visual media processing. VMP-MPSoC adopts a hierarchical 32-bit Wishbone bus. High-speed wishbone bus takes crossbar strategy while low-speed bus takes shared strategy. The synergistic processors are connected to high-speed crossbar through one master and one slave interface. The slave interface allows host processor to configure and access the memory space of synergistic processors, while the master interface allows synergistic processors to move data between host memory and shared memory simultaneously. Fig. 1 shows VMP-MPSoC architecture. VMP-MPSoC has been validated on FPGA board and the MPSoC chip has taped-out under SMIC 0.13um CMOS technology.

VMP-MPSoC includes three PEs-one host processor and two different synergistic processor cores. The application-to-architecture mapping considers the process features of PEs, which mainly represents in the application specific instruction sets, memory access pattern and operands regulation. The host processor has one generic ASIC instruction set; one synergic core has an application specific instruction set that is strong in image and video processing; another synergic core has an application specific instruction set that is good at graphic accelerating. The two cores have both a 4-way SIMD data-path and 64 128-bit width registers. Moreover, they are tightly coupled through a communication and execution scheduler, and they shares one 64-bit local SDRAM memory controller. The host PE can configure and access the memory space of the synergistic PEs, and the synergistic PEs can move data between host memory and shared memory simultaneously. The application-to-architecture mapping makes full use of these features to achieve parallel processing performance and efficiency.

B. Template Regulation

Generally, different heterogeneous MPSoCs are designed with different configurations such as PE configuration, communication protocol, and memory hierarchy to satisfy different requirements. In order to map media algorithms onto heterogeneous MPSoCs, the parameterized architecture template is designed. With different parameter configuration, the template can flexibly represent VMP-SoC architecture in an abstract level and support future product update. The architecture template composes of generic information descriptor (HEADER), specific field feature descriptor (DOMAIN) and architecture descriptor (ARCH), as Fig. 2 shows.

```
//Architecture template of VMP-MPSoC
class VMP-MPSoC-Template
  HEADER  //Generic information descriptor
    name VMP-MPSoC-Template;
    description "XXX";
    path "X:/XX/XX..."
    author "xxx xxx";
    date "XXXX-XX-XX";
  DOMAIN  //Specific application feature descriptor
    Source-code;    //C language
    SUIF-representation;   //SUIF of source code
    PIA-CDTG;       //Application model
    vi-DFG;         //loop nests model
    data-info;      //data info in loop body
    enhanced-data-dependency-graph;  //data dependency
  ARCH  //architecture descriptor
    instruction_sets[m];
    memory_access_pattern;
    operands_regulation;
    constraint_model;
    resource_model;
```

Figure 2. Architecture template of VMP-MPSoC

The HEADER descriptor describes generic information about architecture templates for future update and reuse. It composes of name, brief description, location path, author information and revised date.

The DOMAIN descriptor describes features of specific application fields such as visual media processing field.

Since MPSoC is commonly applied in specific fields, specific application features can give valuable advice for improving MPSoC performance and efficiency in parallel processing. The descriptor composes of C-language source code, SUIF representation, PIA-CDTG model, vi-DFG model, data-info and dependency-graph. PIA-CDTG and vi-DFG model has already been introduced in section 3. The parallel processing begins with reading from the descriptor and achieves optimization according to application features.

The ARCH descriptor describes heterogeneous MPSoC architectures at an abstraction level, which is hardware platform of parallel processing of visual media application. Parallel processing on the MPSoC architecture should have deep and definite acquaintance with the ARCH descriptor. The descriptor composes of instruction sets, memory access pattern, operand regulation, constraint model and resource model. Together with DOMAIN descriptor, ARCH descriptor forms the input of parallel process. As an abstraction of architecture, ARCH descriptor ignores unnecessary details and keeps primary elements for parallel processing, which reduces the problem complexity and helps to address primary issues.

Architecture templates collect all the necessary information for parallel processing and accurately describe them at an abstraction level, which benefits for addressing the parallel processing problem. The parameterized template provides the optimizing exploration space among different pivotal options and supports future product reuse. Moreover, the abstraction projects primary elements in parallel processing to improve MPSoC performance and efficiency.

V. PARALLE PROCESSING METHODOLOGY

In this paper, the proposed approach parallelizes sequential media algorithms on heterogeneous MPSoC using application-to-architecture mapping techniques. And it focuses on critical loops. Critical loops are transformed into parallel tiles according to the dependency between multi-dimension signals in the loop body, which works in the theory framework of tiling transformation proposed in reference [23]. And then tiles are mapping to VMP-MPSoC architecture template for parallel processing.

A. Preliminary

Tiling transformation must subject to the tiling legality, and the definition is given as follows.

Definition 8. (Tiling legality) Tiling transformation forms the set $\Gamma:\{T_1,T_2,...,T_n\}$. The tiling transformation is legal only when the following two conditions are simultaneously met.

- $\forall T_i \in \Gamma$, for $i \in [1,n]$, T_i can independently run on a processor.
- $\forall T_i \in \Gamma$, $\forall T_j \in \Gamma$, for $i, j \in [1,n]$ and $i \neq j$, the execution sequence $< T_1,T_2,...,T_i,...,T_j,...,T_n >$ gets the same results as $< T_1,T_2,...,T_j,...,T_i,...,T_n >$.

When ϕ_{Si} meets the following condition, the hyperplanes and tiling transformation are legal [23].

$$\phi_{sj}(t) - \phi_{si}(s) \geq 0 . \quad (1)$$

Moreover, the following two theorems proposed in Reference [23] will be cited in the process of tiling transformation.

Theorem 1. Let D be a non-empty polyhedron defined by s affine inequalities or faces: $a_k \cdot x + b_k \geq 0$ where $1 \leq k \leq s$. An affine form is non-negative everywhere in D iff it is a positive affine combination of the faces:

$$\psi(x) = \lambda_0 + \sum_k \lambda_k (a_k \cdot x + b_k), \lambda_k \geq 0$$

Theorem 2. If all the iteration space is bounded, there exists an affine function: $v(p) = u \cdot p + w$, which can constrain $\delta_e(t)$. That is $v(p) - \delta_e(t) \geq 0, t \in \mathrm{P}_e, \forall e \in E$ where $\delta_e(t) = \phi_{si}(t) - \phi_{sj}(h_e(t))$ and p is structure parameter vector of loop nests.

B. Mapping candidate generation

Mapping candidate generation are solved in the theory framework proposed by Uday Bondhugula [23].

If $t \in \mathrm{P}_e$ is uniform dependency, the corresponding δ_e is independent to structure parameters, and then w is the upper boundary of δ_e. If $t \in \mathrm{P}_e$ is non-uniform dependency, the upper boundary of δ_e can be solved by ILP as follows.

Firstly, the following equation can be achieved from theorem 1 and 2:

$$v(p) - \delta_e(t) = (u \cdot p + w) - (\phi_{si}(t) - \phi_{sj}(h_e(t))) \quad (2)$$

Equation 2 can be transformed to the form $f(u,w) = g(i)$, which shows that linear inequalities of u and w can be deduced from the inequality $Ai + b \geq 0$. And the linear inequalities can be solved by finding minimum lexicographic order of u and w ordinal. Then the minimum dependency distance is solved, according to which one face of the tile can be obtained by solving u and w under the optimal condition of $\text{minimize}_{\prec}\{u_1,u_2,...,u_k,w,...,c_i^{'}s,...\}$.

One face of the mapping candidate, tile, has been obtained after finding the minimum dependency distance. The next face can be obtained by solving the ILP problem with one more constraint independent to the known solution. The similar work is iterated until n faces have been obtained.

C. Parallel Mapping

As mapping candidates, tiles have already been generated. Each tile can be independently run on a processor. The tiles are parallel mapped onto the heterogeneous MPSoC architecture using eACOGA algorithm [28, 29], which improves on ACO algorithm.

M.Dorigo proposed ACO algorithm in 1991 [17] and gained the prize of Madame Curie Outstanding Achievements for the established sound basis for follow-

up research of ACO. Researches show that the ants with poor vision could find the shortest route from ant colony to food by some volatility secretions namely pheromone. The subsequent ants select this route according to the strength of pheromone. When more ants pass through the route, much stronger the pheromone of the route becomes. Thus more ants are attracted to this route, forming a kind of positive feedback. Existing researches on ACO show well effects on many optimization problems, such as traveling salesman problem, network route, and job shop scheduling. ACO algorithm can find better solutions of partitioning more effectively [30]. But the strategy of random selection in constructing solutions leads to slow convergence speed. Furthermore, the principle of positive feedback can not only strengthen the solutions with better performance, but also bring on the stagnancy of search. The causation is that the main configurable parameters of the algorithm, such as α, β, ρ, Q, are set to fixed value when initializing, and it has no adaptability to various applications.

The proposed eACOGA algorithm for parallel mapping onto heterogeneous MPSoC can evolve configuration parameters of ACO algorithm by cross operation and variation operation. So eACOGA can evolve and optimize itself to search global optimal solutions. The rules of eACOGA are defined as follows:

Objective Function and Fitness Function: The objective function is defined as $S_{best} = \arg\min(1/MF_m)$, fitness function as $Fitness(m) = MF_m$ where MF_m figures the matching value of mapping m.

VMP-MPSOC Architecture Template: As the abstraction of mapping destination, the architecture template figures the necessary details of VMP-MPSoC architecture mapping. As the inputs of the eACOGA algorithm, the template, together with the tiles, provides the mapping exploration space.

Strategy of Render to DAG: For any nodes except v_n, ants try to render the color of v_j, the subsequence of v_i. Ants achieve the work according to the global heuristic information ($\tau_{ij}(k)$) of edge e_{ij} and the local heuristic information ($\eta_j(k)$) of node v_j. The ants on node v_i will render the color of node v_j as c_k at the probability of:

$$p_{ij}(k) = \frac{\tau_{ij}(k)^{\alpha}\eta_j(k)^{\beta}}{\sum_{l=1,2}\tau_{ij}(l)^{\alpha}\eta_j(l)^{\beta}}$$

where $\tau_{ij}(k)$ is the pheromone on edge e_{ij}, α and β is the factor of them and $\eta_j(k)$ is defined as follows:

$$\eta_j(k) = \alpha \cdot I_{i,j}(k) + \beta \cdot D_{i,j}(k)$$

Use Genetic Algorithm to Evolve Parameters: eACOGA algorithm uses genetic algorithm (GA) to evolve the parameters of ant colony optimization, such as α, β, ρ, Q. First, configure the probability factor of cross and variation operation according to the size of population and the generation of evolution. The evolution rules of GA use proportional selection, single cross and even variation. Then by taking α, β, ρ, Q as the variable of fitness function, the best optimal partition cost as fitness function and the course of ACO as the individual, we optimize α, β, ρ, Q repeatedly until finding the best optimal solutions.

Pheromone Setting and Refreshing: eACOGA algorithm adopts MMAS (Max-Min Ant System) introduced by Thomas Stuzle [31], the refreshing equation of pheromone is:

$$\tau_{ij}(k) = \begin{cases} (1-\rho)*\tau_{ij}(k) + \Delta\tau_{ij}(k)_{best} \\ \tau_{ij}(k)_{\max}, \text{if } \tau_{ij}(k) > \tau_{ij}(k)_{\max} \\ \tau_{ij}(k)_{\min}, \text{if } \tau_{ij}(k) < \tau_{ij}(k)_{\min} \end{cases}$$

Where, ρ is the evaporation ratio of pheromone, $\tau_{ij}(k)_{\max}$ ($\tau_{ij}(k)_{\min}$) is maximum (minimum) strength of c_k pheromone on edge e_{ij}, and $\Delta\tau_{ij}(k)_{best}$ is increment of c_k pheromone on edge e_{ij} done by the "best ant" in current ant system algorithm iteration. According to Ant-Cycle Model, $\Delta\tau_{ij}(k)_{best}$ is defined as:

$$\Delta\tau_{ij}(k)_{best} = \begin{cases} Q / C_{Pbest}, in \ \ P_{best} \ \ e_{ij} \ \ is \ \ rendered \ \ by c_k \\ 0, otherwise \end{cases}$$

VI. Implementation and Preliminary Results

As an important part of application development platform for VMP-MPSoC, the tool named PPT-MPSoC (Parallel Processing Tool for heterogeneous MPSoC) is designed. PPT-MPSoC is designed on the basis of the above parallel processing methodology. Fig. 3 shows PPT-MPSoC framework. It parallel processes application onto heterogeneous by the way of application analysis, tiling, tile-to-architecture mapping and mapping space exploration. The auxiliary tools include library management tools, performance analyzer and compiler. The main methodologies involved in the parallel processing are dependency analysis, parallel transformation and eACOGA optimization.

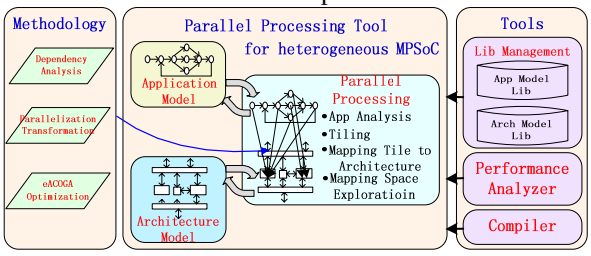

Figure 3. PPT-MPSoC Framework

A group of typical visual media algorithms, list in table 1, is used as the benchmark of PPT-MPSoC. Table 1 shows the performance comparison of the benchmarks running on different conditions. Column 2 in Table 1 shows the execution time of benchmarks sequentially running on ARM926E. Column 3 shows the execution time of benchmarks running on VMP-MPSoC after manual parallel processing. Column 4 shows the execution time of benchmarks running on VMP-MPSoC after parallel processing using the proposed methodology in this paper.

TABLE I.
TYPICAL ALGORITHMS OF VISUAL MEDIA PROCESSING

Alg.	Seq. Exe Time (Cycles)	Exe. Time after Man. Parallel Pro (Cycles)	Exe. after Parallel Processing using the proposed Methodology		
			Time (Cycles)	Acc.Ratio/ Seq. Exe	Acc.Ratio/ Mau.Para.
Sin x4	12/96	2	2	6/48	1
Cos x4	12/96	2	2	6/48	1
Log x4	144	8	6	18	1.3
Exp x4	160	8	6	26	1.3
Integer ALU x8	8	1	1	8	1
Vertex Trans. X8	208	10	4	52	2.5
Light x8	76	4	2	38	2
128-FFT x2	130638	6048	2012	64	3.0
Matrix Mul. 64x64 x2	14460	1126	856	16	1.3

The proposed approach in this paper parallelizes sequential media algorithms on VMP-MPSoC using tiling transformation and application-to-architecture mapping techniques. It focuses on accelerating loop nests. More time loop nests consume in the whole algorithm, more apparent effects the proposed approach makes, as the parallel processing of the 128-FFT algorithm shows. In the tile-to-architecture mapping, tiles with good data locality reduce communication cost and improve processing performance, but if the size of tiles is unfit for the ability of PEs, too small or too big, the performance improvement is little or may recede instead under some specific conditions. Generally, in the field of visual media processing, the proposed approach can obtain approximate or better accelerating than the manual parallel processing by specialized developers.

Ⅶ. CONCLUSION

This paper presents a new approach of parallel processing sequential media algorithms onto heterogeneous MPSoC using program transformation and application-to-architecture mapping techniques. The parallel processing problem is definitely formulated firstly and VMP-MPSoC architecture template is defined to describe target architectures at an abstraction level. As mapping candidates, tiles with data locality are generated by solving ILP problem after dependency analysis. And tile-to-architecture mapping using eACOGA algorithm finds "the optimal" scheme considering the features of tiles and heterogeneous MPSoC. Experimental results show that the proposed approach can obtain approximate or better accelerating than the manual parallel processing by experienced designers within typical benchmarks.

However, the proposed approach has some limits. Firstly, it should be validated using more visual media algorithms. The apprehensive application will give more advices for improving the approach, which is the next step of the research. Moreover, the proposed approach can reduce communication cost indirectly by data locality optimization. The future work will investigate the relation of parallel processing and communication cost and try to further optimize communication cost.

ACKNOWLEDGMENT

This work is sponsored by the National Science Foundation of China under the grant NO.90207019 and NO.90707003.

REFERENCES

[1] S. Hu, "The development of research on visual media intelligent processing," Report on Chinese Development of Computer Science and Technology, Beijing, Tsinghua University Press, 2007,pp. 269-383.

[2] "OpenMP homepage," 2009, http://openmp.org.

[3] "MPI homepage," 2009, http://www.open-mpi.org/.

[4] J. Ceng, J. Castrillon, W. Sheng, H. Scharwachter, R. Leupers, etc, "MAPS: an integrated framework for MPSoC application parallelization," the 45th ACM/IEEE Design Automation Conf., 2008, pp. 754-759.

[5] M. Kulkarni, K. Pingali, B. Walter, G. Ramanarayanan, K. Bala, and L. Chew, "Optimistic parallelism requires abstractions," Proceedings of the 2007 ACM SIGPLAN Conf. on Programming Language Design and Implementation, 2007, pp. 211-222.

[6] T. Wiangtong, P. Cheung, and W. Luk, "Hardware/software codesign: a systematic approach targeting data-intensive applications," IEEE Signal Processing Magazine, vol. 22, no. 3, 2005, pp. 14-22.

[7] L. Thiele, I. Bacivarov, W. Haid, and K. Huang, "Mapping applications to tiled multiprocessor embedded systems," Proceedings of the 7th International Conf. on Application of Concurrency to System Design, 2007, pp. 29-40.

[8] M. Gordon, W. Thies, M. Karczmarek, J. Lin, A. Meli, etc, "A stream compiler for communication-exposed architectures," ACM SIGOPS Operating Systems Review, vol. 36, no. 5, 2002, pp. 291-303.

[9] "Embedded software stuck at C," 2009, http://www.eetimes.com/news/design/showArticle.jhtml?articleID=202102427.

[10] A. Pimentel, C. Erbas, and S. Polstra, "A systematic approach to exploring embedded system architectures at multiple abstraction levels," *IEEE Trans. on Computer*, 2006, pp. 99-112.

[11] A. Pimentel, L. Hertzberger, P. Lieverse, P. van der Wolf, and F. Deprettere, "Exploring embedded systems architectures with Artemis," Computer, 2001, pp. 57-63.

[12] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli, "Metropolis: an integrated electronic system design environment," Computer, 2003, pp. 45-52.

[13] M. Ruggiero, A. Guerri, D. Bertozzi, M. Milano, and L. Benini, "A fast and accurate technique for mapping parallel applications on stream-oriented MPSoC platforms with communication awareness," International Journal of Parallel Programming, vol. 36, no. 1, 2008, pp. 3-36.

[14] E. Carvalho, N. Calazans, and F. Moraes, "Heuristics for dynamic task mapping in NoC-based heterogeneous MPSoCs," Proceedings of the 18th IEEE/IFIP International Workshop on Rapid System Prototyping, 2007, pp. 34-40.

[15] P. Paulin, "Automatic mapping of parallel applications onto multi-processor platforms: a multimedia application,"

Euromicro Symposium on Digital System Design, 2004, pp. 2-4.

[16] M. Yan, S. Li, J. Shen, L. Xie, L, Liu, G. Wan, and L. Yin, "A heterogeneous multicore SoC optimized for embedded visual media process," Proceedings of WRI International Conference on Communication and Mobile Computing, vol.3, 2009, pp 12-16.

[17] M. Ruggiero, A. Guerri, D. Bertozzi, F. Poletti, and M. Milano, "Communication-aware allocation and scheduling framework for stream-oriented multi-processor systems-on-chip," Proceedings of the Conf. on Design, Automation and Test in Europe, 2006, pp. 3-8.

[18] K. Vivekanandarajah, and S. Pilakkat, "Task Mapping in heterogeneous MPSoCs for system level design," The 13th IEEE International Conf. on Engineering of Complex Computer Systems, 2008, pp. 56-65.

[19] P. Mudry, G. Zufferey, and G. Tempesti, "A hybrid genetic algorithm for constrained hardware/software partitioning," The 2nd International Conf. on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006, pp. 1–6.

[20] C. Erbas, S. Cerav-Erbas, and A. Pimentel, "Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design," IEEE Trans. on Evolutionary Computation, vol. 10, no. 3, 2006, pp. 358-374.

[21] P. Zhao, S. Li, D. Wang, and M. Yan, "Application-driven System-on-Chip system model extraction approach," The 12th International Conf. on Computer Supported Cooperative Work in Design, 2008, pp. 123-128.

[22] D. Suresh, W. Najjar, F. Vahid, J. Villarreal, and G. Stitt, "Profiling tools for hardware/software partitioning of embedded applications," ACM SIGPLAN Notices, vol. 38, no. 7, 2003, pp. 189-198.

[23] U. Bondhugula, M. Baskaran, S. Krishnamoorthy, J. Ramanujam, A. Rountev, and P. Sadayappan, "Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model," Lecture Notes in Computer Science, vol. 4959, 2008, pp. 132.

[24] F. Irigoin, and R. Triolet, "Supernode partitioning," Proceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 1988, pp. 319-329.

[25] R. Wilson, R. French, C. Wilson, S. Amarasinghe, J. Anderson, etc, "SUIF: an infrastructure for research on parallelizing and optimizing compilers," ACM SIGPLAN Notices, vol. 29, no. 12, 1994, pp. 31-37.

[26] S. Khuri, T. Back, and J. Heitkotter, "The zero/one multiple knapsack problem and genetic algorithms," Proceedings of the 1994 ACM Symposium on Applied Computing, 1994, pp. 188-193.

[27] S. Shen, "EStar documents," National University of Defence Techonology, Technique report, unpublished, 2003.

[28] D. Wang, S. Li, and P. Zhao, "Hardware/software partitioning approach of coarse-grained reconfigurable system using ant colony optimization," Computer Journal of China, in press, 2009.

[29] D. Wang, S. Li, and Y. Dou, "Collaborative hardware/software partition of coarse-grained reconfigurable system using evolutionary ant colony optimization," Asia and South Pacific Design Automation Conf., 2008, pp. 679-684.

[30] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents,", IEEE Trans. on Systems, Man and Cybernetics, Part B, vol. 26, no. 1, 1996, pp. 29-41.

[31] T. Stützle, and H. Hoos, "Max-min ant system," Future Generation Computer Systems, vol. 16, no. 9, 2000, pp. 889-914.

Peng Zhao was born in Henan, China in 1979. In 2004, he obtained his MS degree in Computer Science from National University of Defense Technology, Changsha, Hunan, China. Since 2005, he has been a PhD candidate at Computer School in National University of Defense Technology. His interests are centered on System-On-Chip system-level design methodology and System-on-Chip performance optimization.

He has attended projects mainly including embedded micro-processor and visual media processing System-On-Chip design, which is sponsored by the National Science Foundation of China. He has got about 12 publications. Current and previous research interests focus on System-on-Chip performance optimization.

Dawei Wang was born in Liaoning, China in 1980. In 2001, he obtained his bachelor degree in Computer Science from National University of Defense Technology, Changsha, Hunan, China. Since 2004, he has been a PhD candidate at Computer School in National University of Defense Technology. His interests are centered on System-On-Chip hardware/software co-design methodology.

Ming Yan was born in Hunan, China in 1981. In 2005, he obtained his MS degree in Computer Science from National University of Defense Technology, Changsha, Hunan, China. Since 2006, he has been a PhD candidate at Computer School in National University of Defense Technology. His interests are centered on System-On-a-Chip architecture design.

Sikun Li was born in Shandong, China in 1941. He is a professor at the Computer School of National University of Defense Technology. His interests are centered on VLSI design methodology, Electronic Design Automation, virtual reality and visualization.

He has conduct more than 15 national and military projects. He has won more than 11 national and military prizes. He has published more than 42 research articles focused on VLSI design methodology, Electronic Design Automation, virtual reality and visualization.

Prof. Li serves as a senior fellow of China Computer Federation.