ISSN: 2583-2621

Object Tracking Algorithms and Implementations on FPGA's RCS architectures: A Review

Manoj Pandey
Department of Electronics & Communication Engineering
Sarala Birla University, Ranchi, India
manoj.pandey@sbu.ac.in

***Abstract*— Tracking a moving object in real-time is an active field of today's machine vision application. Tracking a moving object/s in real-time demands an efficient tracking algorithm and competent hardware/software platforms for implementations. Traditionally, tracking is computationally intensive task due to the processing requirement of high volume of data. However, the capability of Field Programmable Gate Array (FPGA) allows them to use in a variety of applications due to its nature of partial and dynamic reconfiguration capabilities. This paper presents the review of tracking algorithms and FPGA's Reconfigurable Computing System (RCS) designs for object tracking over last two decades. Although, many survey exists on tracking algorithms but the survey on its implementations addressing to FPGA's and RCS architectures is missing. We believe this survey would fill the gap of recent tracking algorithms and its hardware/software implementation and complete the view of existing designs. It is also expected to convey the future directions to the researchers can follow in this field in coming years.**

***Keywords*— Object tracking, FPGA and reconfigurable computing**

I. INTRODUCTION

Image/video processing in real-time is the essential need of modern machinery era. The applications e.g. surveillance, driver assistance, air traffic control, medical imaging, humanoid etc. are the need of modern automation. To design a robust tracking system in real-time, there is a need to process high volume of image/video data. Adaptability of systems with change of application is also a majorly demanding characteristic of modern application area of machine vision like in automatic target recognition and tracking. Object tracking is defined as the identification or estimation of trajectory of a non-rigid object/s in a sequence of video frames in a scene. With respect to location of the object in current frame, new location is updated in the coming sequence of frames. It employs intensive computation for extracting the object's features in order to extract the required information from large volume data contained in an image. Figure 1, depicts the basic building blocks of object tracker. For any video/image processing task, capturing the image is foremost requirement that is possible by any image sensors or cameras. In most of the tracking task only a small region of image is allocated for tracking, while, rest of the region is assigned to pre-processing. Here, pre-processing defines the correction of pixels either by normalizing or thresholding methods. Before applying the process of object detection or tracking, information about the object such as shape, size, color or location etc. is necessary by which objects are represented or selected. From the selected region or pixels of captured frame, features are extracted for object detection and tracking.

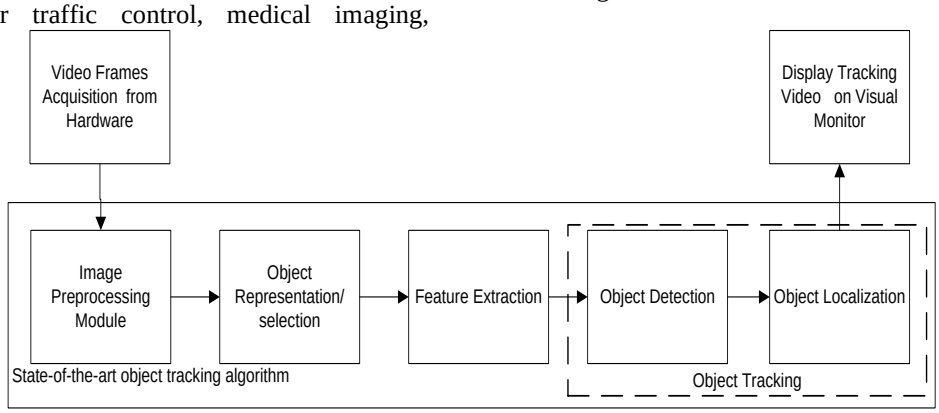

Fig. 1. Basic building block of object tracking system.

Object detection and localization are two major steps to be followed for the tracking of non-rigid objects. The ways by which these two steps are combined and weighted are application dependent that make a role in the in designing of robust and efficient tracker.

With the progressive multimedia standards, the demand of developing a robust visual tracking system has increased many

ISSN: 2583-2621

folds. Executing the complex algorithms for object detection and tracking in real-time have been really challenging for design engineers because of the fast processing requirements over a high volume of data. Traditionally, application specific integrated circuits (ASICs), graphic processors (GPs) and general purpose processors (GPPs) have been preferred to solve the tasks. A reconfigurable hardware based on FPGA technology have shown the tremendous benefits over ASICs, GPs and GPPs. Taking the advantages of partial/dynamic re-configuration of FPGAs, it become more powerful for computing the real-time task. Though, the first idea of reconfigurable hardware based on FPGA technology came in sixties but the first example of configurable computing system appeared in late eighties [1]. There have been extensive research into FPGA exploited reconfigurable computing system (RCS) architectures for embedded applications of signal processing, traffic monitoring, image/video processing and robotics etc. Thus the reconfigurable computing is defined as the ability to execute the functional task on FPGA hardware to increase overall performance of systems retaining the flexibility of software. For last two decades FPGAs have proved its capability in embedded architectures for various applications e.g image/video and machine learning processing as well. Performance with flexibility & adaptability are key features reconfigurable computing which offers cost-effective solution for computationally intensive real-time application through the reuse of reconfigurable fabric hardware. Using partial & dynamic reconfigurations, RCS become more capable in terms of hardware utilization and also improve the adaptability of system with change in application. RCS often has given impressive performances in term of execution speed, resource saving and efficient in energy savings. Most FPGAs support reconfiguration to provide higher flexibility without losing its performance [2].

For instance, an optical flow algorithm by Javier et al. [3] is implemented on both on software as well as on hardware. The author declared his observations as: 30 frames per second (fps) frame rate and 5% less execution time for 320x240 image dimension. In other hand, the hardware cost is reported as: 21,649 FPGA slices. In the sequence of similar work reported by Kristensen et. al. [4] for real time surveillance system. A segmentation and morphological functions are implemented on FPGA hardware for image dimension of 320x240 pixels. Author declared in his observation as: 25 fps of frame rate & 70% memory reduction. Software implementation FPGAs in other hand stands in concurrent as hardware. FPGA vendors (e.g Xilinx Inc. & Altera), now in the days are providing more configurable soft-core as well as hard-core processors that can be used to synthesize the algorithms on their FPGAs itself [5]. Therefore, both the parallel/sequential processing [6-7] can be used for hard-ware/software implementation for high speed real-time video/image processing embedded architectures [8-10]. It was recognized in 1989 by Gray and Keen [11] that many image processing algorithms would perform well on reconfigurable computers for the purpose of image processing algorithms.

This article is compiled in six sections. Section 1, summarize an introduction of object tracking systems and implementation approaches. Section 2, illustrates the review of object tracking flow along with state-of-the-art tracking algorithms. Section 3, covers the basic building block of logical functions along with some of the popular FPGA's RCS architectures. Finally, section 4 conclude the article.

II. Tracking Flow and Algorithms

In today's scenario, new algorithms are evolving quickly. Many of these algorithms describe a parametric/non-parametric statistical description of the object representation. As we know, parametric statistical method is a nonlinear estimation method while nonparametric are based on unpredictable to the probability distributions of the variables being assessed. As we know, in parametric statistical analysis the targeted object/location is generally fitted in Gaussian model, while in non-parametric, estimation is done by training of random samples/data. A detail architectural process of object tracking is presented in Figure 2, that comprises various steps: i) image acquisition, ii) image preprocessing, iii) object representation, iv) feature extraction, v) object detection, vi) object tracking and vii) tracking validation/displaying. Each steps have various methods and corresponding to these methods various algorithms have evolved to process the images. Since to discuss each algorithms for all processing step is tedious task therefore, the methods/algorithms used for object detection/tracking are mainly presented in this section. Trackers are classified mainly in three categories: point tracking [12], kernel tracking [13] and silhouette tracking [14]. Point tracking is again categorized into deterministic and probabilistic (statistical) methods. Modified Greedy Effect (MGE) tracker [15] and Kalman filter [16] are examples of deterministic and statistical method respectively. Kernel tracking and silhouette tracking both are further classified in two classes like as kernel tracking is classified as multi view and template based [17]; whereas silhouette tracking is classified as: contour evaluation [18] and shape matching [19]. To meet the tracking environment or constraint, a selection of appropriate algorithm is important. Sometime tracking becomes complex due to noise in images, varying in shapes and size, complex object motion, scene illumination changes, object occlusions or real-time processing requirements that causes loss of information from images [12]. A designer can simplify the tracking by applying the constraints such as motion, shape or appearance of objects. For example, point tracking is used in deterministic and statistical methods of tracking.

In deterministic approach, a MGE is used in tracking by maximizing the rigidity and proximity constraints. It has shown promising ad-vantages in minimizing the computational cost. However, this approach is unable to handle the occlusions. In other context, a statistical approach used for tracking where a position and velocity of object are predicted and new positions are estimated based on Kalman filtering[16] techniques. This method has shown promising advantages for tracking but strictly based on Gaussian distribution.

ISSN: 2583-2621

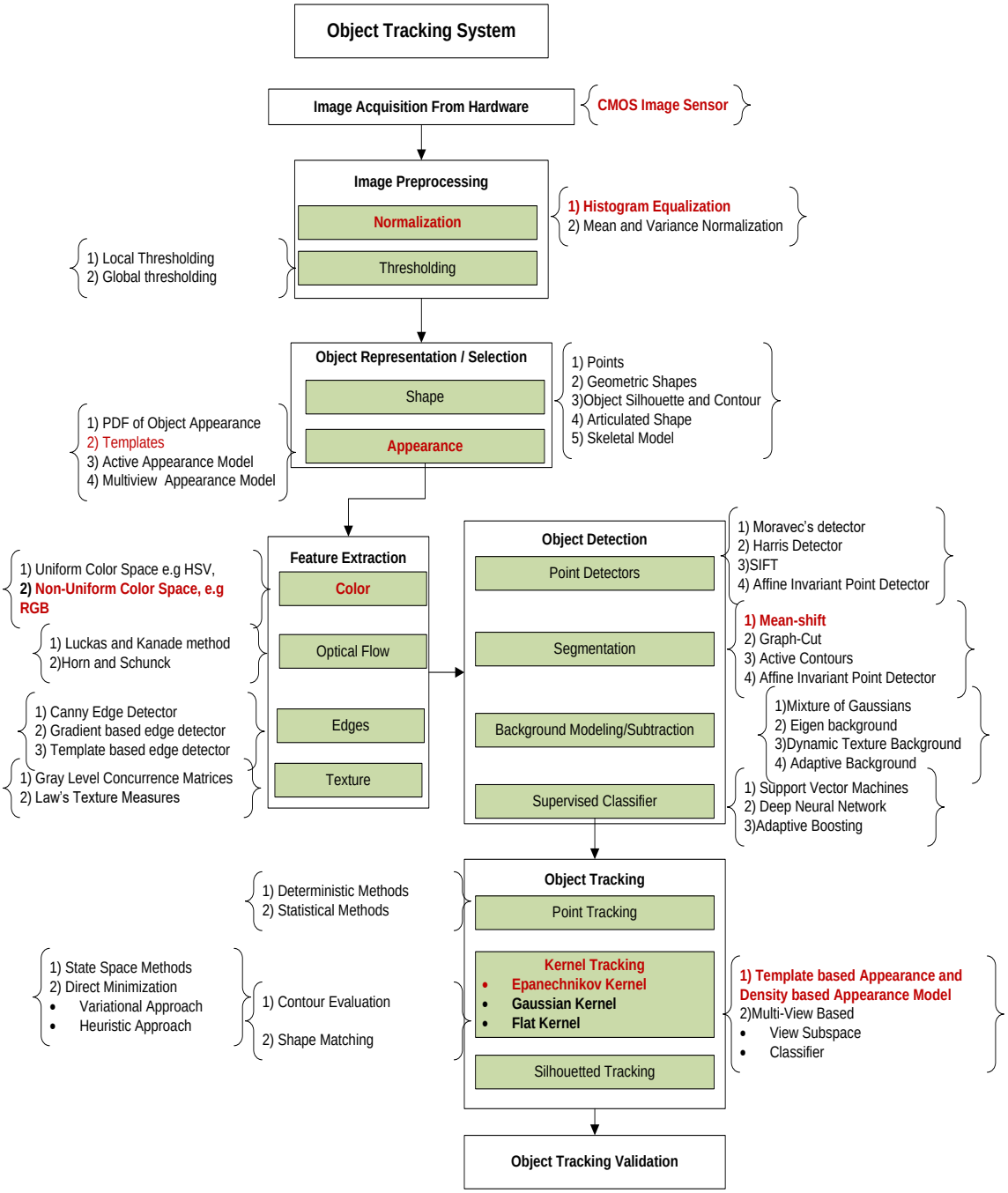

Fig. 2. Typical object tracking flow; highlighted with red color, represents the flow of kernel based mean-shift object tracking

There are various object tracking algorithms which have evolved in last decades as mentioned in Table. 1. Many of these algorithms are either region-based [20] or appearance-based [21] approaches. Many of the articles have described these approaches. A training based tracking system in other hand is like deep Neural Network (deep NN) [22] that uses to search the targeted object in sequence of frames. Other side, a model-based tracking methods are dependent on pre-initialized data that tar-get objects based on the features of: shape, skin, silhouette, skeleton, intensity, color, edges and contours etc. These features are used to establish a relation/similarity between candidate and target models. A detail tracking process as displayed in Fig.2, represent the tracking algorithms by using kernel based means shift algorithms [13]. Generally, a kernel density estimation is one of the popular non-parametric ways to estimate the statistical correspondence with the help of randomly distributed samples/variables. Additionally, in a given data samples, an appropriate kernel selection and density

ISSN: 2583-2621

estimation works well in both low and high dimension and applied in tracking applications [23][24].

TABLE 1. OBJECT TRACKING APPROACHES.

Algorithm	Feature Selection	Application	Advantages/Limitations
Adaptive Block matching[25]	Using object contour, motion vector, object boundary	MPEG-4, Videos	Computationally superior, good in compression
Eigen Tracking [26] [27]	View based Eigen based space representation	Nearest view is recovered, gesture recognition	Useful for view point and changes in pose.
Mean shift[28][29]	Color PDF	Moving Object, smoothing and segmentation	Good for Cluster analysis, global optimization and performance
Kernel based Mean Shift[30][31]	Kernel: Gaussian/ Epanechikov Kernel	Real time tracking	Good tracking for uniform, single and slow moving objects
CAM-shift combination with difference in frame[32]	Difference in frame movement region, Motions segmentation	Motion Estimation	Shortcoming of artificial orientation and divergence
Deep Neural Network [33][34][35]	Training de-noising auto encoder	Visual Tracking	Unsupervised method, newly evolved algorithm Room available to improve performance
SIFT Features and Mean Shift [36]	Color features	Real Time Object Tracking	Improved Performance Than Classical Mean Shift
PCA [26]+ ICA [37]	Basis image, generalization of PCA	Face Recognition	Better face recognition
Adaptive Kalman filtering[38]	HSI Color Space	Moving Object Tracking in a Video	Less computational cost than Kalman filter Limits to constant velocity or constant acceleration
Image correlation[39]	Kalman predictor adaptive optimized matching	Object detection and tracking	Decreases computational cost and Show better performance Difficult to realize in real time
Particle filter [40] [41]	Color based Particle filter and Kalman Filter	Shape similarity and shape information	Good performance in shape matching and better in real-time.
Background subtraction statistical[42,43]	Modeling and updating background, illumination changes and shadows	Object Detection	Fast and effective
Canny edge detector Statistical 4 degree of freedom[44,45]	Deformed object	Robotics	Accurate with occlusion and spurious edges
Using stereo vision and sensor information[46]	3D moving object	Tracking	Fast movement and precise tracking
Diamond Search [47]	Block Matching	Motion Estimation	Fast Motion Estimation

The commonly used object tracking algorithms based on the above discussed approaches as reported in Table 1 are: adaptive block matching [25], Eigen tracking [26][27], mean shift [28][29], kernel based mean-shift [13][30][31], Cam-shift [32], Deep Neural Network (deep NN) [33][34] and Scale Invariant Feature Transform (SIFT) [35]. Moreover, there are some of the filtering based trackers such as: Kalman filter [38], image correlation [39], particle filter [22], background subtraction [42] [43], canny edge detector [44][45], stereo vision [46] and optical flow [47] that have been used either independently or mapping with other algorithms to improve tracking accuracy or efficiency.

Eigen tracking [27] and adaptive block matching [25] are developed on view based representation model. In Eigen tracking [26], the design is based on Eigen-space representation model. It is useful in view based representation that can be used to track objects like human hands and change in pose etc. Tracking using block matching [25] is an algorithm which works on pre-estimation of the object boundaries by computing the motion vectors and by updating the contour using occlusions detection.

ISSN: 2583-2621

In addition, there are some other important issues, which are necessary to be considered in proper detection of moving object. The localization and segmentation are two important task for through which reconstruction and recognition can take place with-out the appearance of the background. The third task as reported in literatures is the transformation of input image into some canonical form through which a similarity can be established. A framework presented in [42], provide a local search method that is robust to background variation. Simultaneously this approach becomes more useful in translation, rotation, and scale in Eigen space and image. Compared to existing tracking algorithms this method is non-invariant to image transformations. Some of these non-invariant parameters are e.g translation, scaling, rotation, occlusion, background clutter, noise etc. The visual-based algorithms are generally used in humanoid applications. Tracking the articulated objects: e.g. human hands, undergo for the changes in both viewpoint as well as changes in position. In some of the cases, it may not perform better in applications where actual object recognition is required. Thus algorithm based on edge tracking [44] for the development of parallel vision system has been used to avoid complexity at several levels and are dependent on parallel processing for rapid computation.

Here, the approaches discussed for tracking have been used either independently or associated with some other filtering or smoothing methods to improve the tracking efficiency and accuracy. For example, a mean shift combined with motion vector analysis [47], an edge detection based on fast adaptive mean shift [28] and kernel based mean-shift integrated with SIFT [36] have been employed for same application. A mean-shift combining with motion vector analysis is developed to improve the tracking efficiency. Similarly, a mean-shift is used in combination of kernel function for the tracking of moving object [24]. In [36], a scale invariant feature trans-form (SIFT) features are used to search a similarity between the Region of Interest (RoI) across a sequence of video frames. In other hand, a mean-shift is applied to conduct similarity search via color features. However, by integrating kernel and mean-shift which are used for smoothing and correspondence establishment between the RoI across frames respectively improves the tracking efficiency and consistent performance. Further, a particle filter [40][41] has also been associated with mean shift algorithm. This method has overcome the large amount of calculation in mean-shift. It is not only useful in real time but also having the facility of high detection probability even in situation of occlusions. Therefore, a feature-based detection or tracking approaches in association of other filtering or smoothing functions have shown promising advantages and improved the accuracy.

III. Review of FPGA/RCS Implementation of Object Tracker

Formerly, the application of computer vision has been limited due to lack of computing capability of high volume of data complexity. Since last decades, comparative to ASIC and SW, reconfiguration techniques based on FPGA technology have given promising benefits in terms of speed, flexibility and power utilization. FPGA which is full of reconfigurable fabric has become the most common device preferred by embedded designers for real time applications in recent years. A reconfigurable fabric represents the logical hardware in which the functionality of logic gates or logical functional blocks are customized at rum time. The connection between these logic gates are also possible to configure or reconfigure. RCS as a whole depends on reconfigurable fabric available in FPGAs and controllers to exploit the capabilities of FPGAs. The reconfigurable fabric executes the task of hardware and controller is used to reconfigure hardware logic and is responsible for function such as external I/O communication. Controllers are used to control the operations either on hardware through HDL (hardware description language) or on SW (on processor). To apply the reconfiguration for a set of application the whole FPGA area is partitioned in reconfigurable region and static region (see Fig 3). The functionality of reconfigurable region is altered by configuring with partial bit files PR1.bit, PR2.bit, PR3.bi etc. and full bit file keeps the information of complete design. Functionality of static region re-mains unaffected during reconfiguration of partial bit files on reconfigurable region.

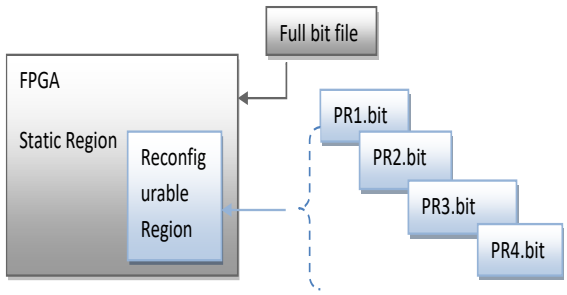

Fig. 3. Partial reconfiguration showing the configuration of full bit file and partial bit file

Xilinx 6200 FPGA [48] was an early partially reconfigurable device where each logic block can be programmed independently. The FPGAs such as Atmel AT40K [49], Xininx Virtex II/II pro, Virtex 4/5/6/7, Kintex-7, and Zynk-7000 are some of the popular PR supporting boards which can be realized by the available PR design sup-porting tools such as Xilinx PlanAhead and Xilinx ISE 12.1 or its higher version.

A. Fine/Coarse grained Architecture

In general, the reconfigurable computing system architectures can be broken down in smallest functionality units (i.e reconfigurable processing fabrics) which are known as granularity. Granularity measures the amount of functionality encapsulated by a region. They range from fine-grained to coarse-grained fabrics. Both the architectures fine/coarse grained can be designed as coprocessor architectures type. In fine-grained RPFs, the data are processed at bit level while in coarse-grained fabrics the data are processed as a group of bits through large functional units such as ALUs/multipliers.

ISSN: 2583-2621

A fine-grained are used on individual bit manipulations without wasting other reconfigurable resources. For complex arithmetic or logic operations, fine-grained pro-cessing elements are used to implement a basic operation. The most common functional units of fine-grained systems are lookup tables, which are used to compute the bulk of the logic in FPGAs. Generally, commercial FPGAs contain many three to six input LUTs and each of these LUTs can be thought as fine-grained functional unit. These fine-grained architectures have significantly more area, power, delay and pow-er overhead. As shown in Table 2, the popular fine grained architectures are: Celoxica RC2000 [50], Xilinx Virtex II pro [51], Altera Stratix II [52], Altera Excalibur [53] and GARP [54]. Custom reconfigurable architecture in general targets the acceleration of software. It means standard PCs can be closely-coupled with the FPGA based accelerator to the computer expansion bus.

TABLE 2. OBJECT TRACKING APPROACHES.

Architecture Name	**Granularity Type**	**Basic Processing Component**	**Routing Architecture**	**Embedded Memory**
Celoxica RC2000 [50]	Fine	4-input LUT	Horizontal and vertical buses	152 MB shared memory
Xilinx Virtex II pro [51]	Fine	4 input LUT	Horizontal and vertical buses	18 kbit blocks
Altera Stratix II [52]	Fine & Coarse	8-bit adaptive logic module	Horizontal and vertical buses	512 kbit blocks
Altera Excalibur[53]	Fine	4-input LUT	Horizontal and vertical buses	2 kbit
GARP[54]	Fine	Logic/arithmetic function	2-bit buses in horizontal	External to fabric
MORPHOSYS[55]	Coarse	ALU and Multiplier	Buses	External to fabric
REMARC [56]	Coarse	ALU	PLB	External to fabric
ADRES [57]	Coarse	ALU	PLB	External to fabric
PIPERENCH [58]	Coarse	8-bit ALU	8-bit buses	External to fabric
RAPID[59]	Coarse	ALUs	Horizontal and vertical	Embedded memory blocks
ReconOS [60]	Heterogeneous	Basic CLBs	System bus & PLB	Embedded memory blocks
SCORE[61]	Pipe-and-Filter	Basic CLBs	System bus & PLB	External SDRAM
SPREAD [62]	Hw/Sw Co-design	Basic CLBs	System bus & PLB	External SDRAM
MOOSE [63]	Hw/Sw Co-design	Basic CLBs	PLB	Embedded memory blocks
RCADE [64]	Hw/Sw Co-design	Basic CLBs	PLB	Embedded memory blocks

Examples are: Garp [54], Morphosys [55], REMARC [56], and ADRES [57]. Garp [54] and Morphosys [55] are example of closely coupled coprocessor architectures where Garp is a fine-grained and Mor-phosys comes under the coarse-grained. The very popular fine-grained architecture designed by BRASS research group is the Garp reconfigurable processor [54] that is MIPS processor and on-chip cache combined with an RPF. Garp [54] as in Figure 4, combine single-issue MIPS (million instructions per second) processor with reconfigurable hardware and it is used as an accelerator. It is consisting of two-dimensional array of logic blocks interconnected by programmable wiring. It has four 32-bit data buses and one 32-bit address bus. The Instruction Set Processor (ISP)-MIPS coupled with the system is capable to move data in and out from array registers. During the operation into array blocks, it has direct access control with the memory blocks. The efficiency of fine-grained designs can be improved by adding architectural support functional blocks.

ISSN: 2583-2621

A coarse-grained system on other hand, are much larger and consist of ALUs, multi-pliers and the memories. Examples of coarse-grained architecture are PipeRench [58] and Rapid [59]. The PipeRench RPF architecture is an ALU based architecture

In contrast to PipeRench; RaPiD architectures can be used as independent of coprocessor or sometimes inte-grated with others computing blocks. RaPiD is specifically designed for the applications where repetitive pipelined computations are required like in nested loops. Most of the designers use hardware design paradigm for programming in configurable

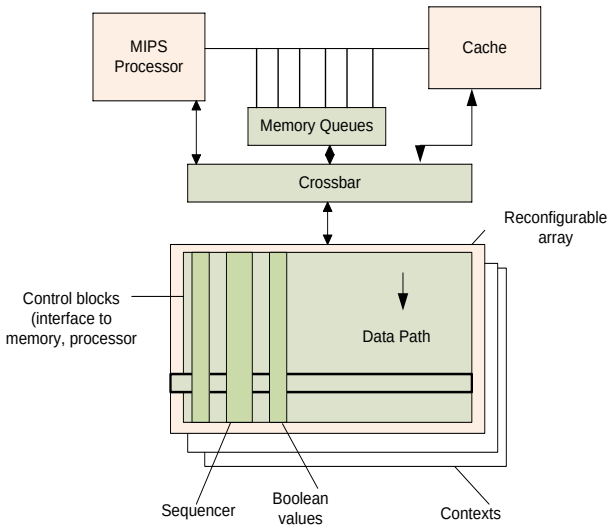

Fig. 4: Fine grained architecture: GARP [54]

In coarse-grained architectures the data are processed as a group of bits through large functional units such as ALUs/multipliers. Coarse-grained is relatively a large number of logical functions contain in a single logical block than fine-grained. For example, 4/5-input LUTs, four multiplexer, and some other fast logic functions may contain in a logic block. The popular examples of coarse-grained architectures are: MorphoSys [55], REMARC [56], ADRES [57], PipeRench [58] and RaPid [59]. In PipeRench [58] for example, RPF architecture is an ALU based that is used as a coprocessor coupled with a host processor. In PipeRench, RPF uses pipelined configuration that works as partially and dynamically reconfigurable. In contrast to PipeRench [58]; RaPiD [59] architectures can be used as independent of coprocessor or sometimes integrated with other computing blocks. RaPiD [59] is particularly designed for the applications where repetitive pipelined computations are required like in nested loops Morphosys [55] as in Figure 5, is one of poplar coarse-grained coprocessor architectures consist of: 8×8 array of logic cells, RISC processor, context memory, frame buffer, cache and frame buffer. Each cell contains register, shifter, ALU, multiplier and register file. The operation performed by each logic cell is defined by a context register (like a configuration memory) and the context memory through which con-tent is used as a coprocessor to a host processor. In PipeRench, RPF uses pipelined configuration that works as partially and dynamically reconfigurable system.

are-as but speedup of execution has been a major challenge for them. Execution in con-figuration system can be speedup up to some extent by extracting loops, frequently used conditions and arithmetic expression from the codes and map them in configurable hardware.

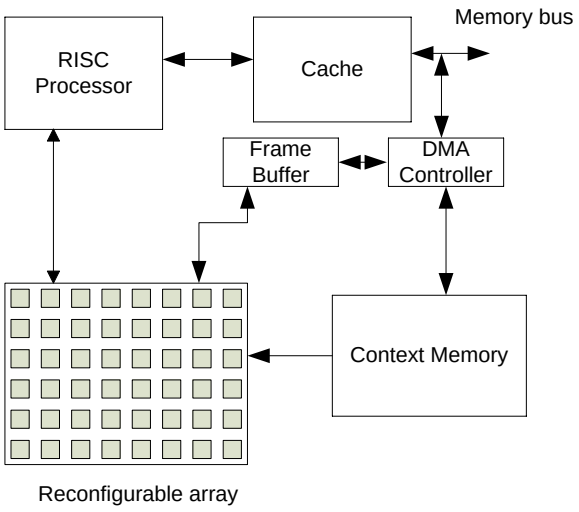

Fig. 5: Coarse grained architecture: MorphoSys [55]

loaded at each execution cycle. The MorPhosys system thus uses common memory context for each cell in the array and implements a single-instruction multi-ple data (SIMD) machines, where the executing function is configurable and depends on the context word.

However, as a disparity between fine and coarse granularity is that, fine-grained consumes more programmable interconnects compared with coarse-grained. It is be-cause as the granularity of logic blocks increases, consecutively the amount of connections into the logic blocks decreases compared to the function they can support. A coarse-grained system, are much larger and consist of ALUs, multipliers and the memories. The current commercially available FPGAs are embedded with more coarse-grained functional blocks. The common coarse-grained blocks available in modern FPGAs are memory block, multiplier digital signal processing (DSP) blocks, processor blocks and some more embedded blocks. In term of speedup performance, power saving and area utilization, coarse-grained RPFs are better choices than its counterpart fine-grained. As discussed above, GARP [54] and MorphoSys [55] are a closely-coupled coprocessor architecture in which reconfigurable hardware augments the instruction set processor as a coprocessor or even being integrated into the proces-sor execution path.

ISSN: 2583-2621

B. *HETEROGENEOUS ARCHITECTURE*

To obtain better improvement in term of performance and reduced area, a heterogeneous architecture i.e. ReconOS [60], gives a thread-level adaptability of embedded systems in both SW and HW threads. ReconOS [60] as shown in Figure 6, consists of hardware and software level multiple threads. Software and operating systems run on a master CPU and one worker processor and two dynamically reconfigurable hard-ware slots are connected to the master processor using a common shared bus. In the architecture, operating system interface (OSIF) is a dedicated hardware, which man-ages the low-level synchronization and includes the necessary logic for partial recon-figuration. Master CPU executes the operating system kernel (OS kernel) and controls all the operations between each thread.

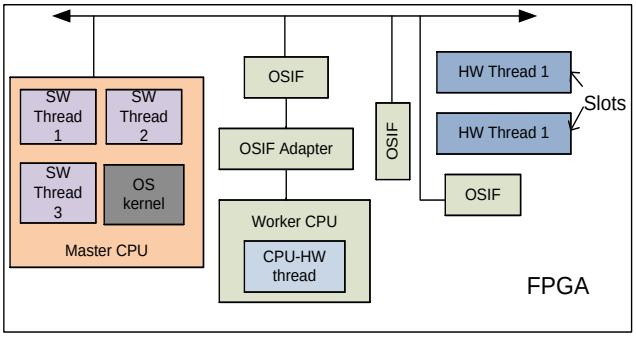

Fig. 6: Heterogeneous architecture: ReconOS [60]

The ReconOS presents a flexible foundation to design a Sequential Monte Carlo (SMC) application for embedded heterogeneous multi-core systems. The ReconOS is capable to repartition of hardware/software composition of real-world tracking application to meet the predefined soft real-time constraints in video stream.

C. *HARDWARE/SOFTWARE CO-DESIGN*

Large application implementing on microprocessor may require less design effort and more size efficient but at the expense of performance. In other context, standalone hardware based system meets design performance but in cost of flexibility and de-sign. Besides the characteristics behaviour variations between a hardware and soft-ware, a hardware/ software co-design is a unified view of design flow for complex systems, which includes simultaneous and separate developments of hardware and software. It is a top-down approach, which includes the identification of tasks, partitioning and implementing the task on hardware and software separately and finally developing inter-task interfaces and communication. The problems occurred during integration, change is very difficult either in hardware or in software. To solve this problem, system-level hardware/software co-design architecture have introduced that eases the integration and detects the error in hardware and software both.

SCORE by Caspi et al. [61] is a programming model, which combines pipe-and-filter architecture, customized compiler and run-time support but it has not been implemented on FPGA platforms. Similar to this, SPREAD by Happe et al. [62] is a FPGA based reconfigurable architecture with a unified hardware/software multithreaded interface and a high throughput streaming structure. The SPREAD [62] architecture as shown in Figure 7, is designed for cryptographic application is a streaming-based partially reconfigurable system which consists of a CPU, reconfigurable processing units, external SDRAM, a configuration controller and some other peripherals. CPU is used to run software threads and operating system. RPUs on other hand are coarse-grained i.e. used for hardware threads and can be directly configured, switched and terminated during run-time. SPREAD architecture is capable to run streaming applications with a unified view of threads. The streaming computation task executes on hardware/software threads. The task T1 and T5 shown in a circle is executing on software thread while the task T2, T3 and T4 are executing on hardware thread. Apart from the simultaneous access issue, a streaming path eliminates dataflow bottleneck commonly found in a system bus, making it simple partially reconfigurable architecture suitable for video streaming. It out performs 1.61-4.59 times higher in terms of power efficiency than their implementation on graphic processing units.

Ideally, hardware/software co-design applications are initially modeled as an abstract form without separating the task on hardware or software. This helps the designer to concentrate on algorithmic and logic development rather than implementation de-tails. Higher degree of abstraction helps the designers to develop the model and simulation more quickly. System-C, is C++ library for modelling hardware systems. System-C library has layers of increasing abstraction level. Some of the other modelling environment for reconfigurable systems are MOOSE [63] and RCADE [64].

ISSN: 2583-2621

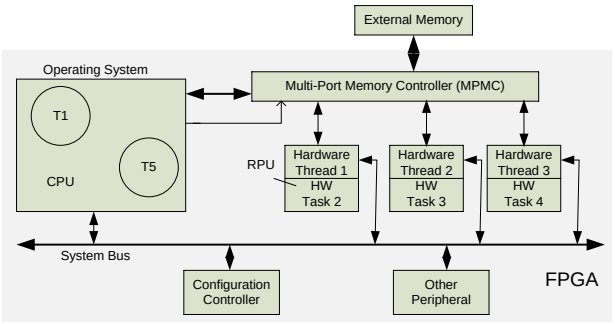

Fig. 7 (a): SPREAD Architecture [62].

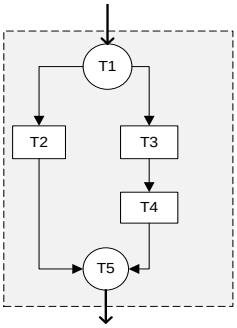

Fig. 7 (b): SPREAD Architecture: streaming task [62].

Traditionally a modular design flow method is used to enable partial and dynamic recon-figuration. Using modular design flow, the first step is design entry either by schematics or by HDLs. Before partial reconfiguration, its functionality and implementation are required to verify. Then only a successful simulation and synthesizing of the design can be started. Although, reconfigurable hardware has shown significant benefits for some applications, but the use of software design environment aids in the creation of configuration for the reconfigurable hardware. Complex algorithm, which is used to process images and image sequences make necessary to do simulations of his operation to verify the fulfillment of the specifications under which it has been de-signed. To test systems that process images it is necessary to feed the model with a complete set of signals and data. Current commercial VHDL synthesis and simulation tools are equipped with utilities for create stimulus signals, but they have limited utility in applications that require a big amount of data like in video/image processing. One of the most useful advantages of VHDL is its capacity to be used in the design of test benches that generate the required signals to stimulate the system under test.

IV. Conclusion

In this paper, we present an existing state-of-art object tracking approaches and how the tracking accuracy are improved by their realization on reconfigurable hard-ware like FPGAs. The objects are mainly represented as: point correspondence, geo-metric models and contour representation. Based on these approaches, many track-ing approaches exist that are required object detection or initialization when the object first appears in a scene. To improve the tracking accuracy even in occlusion or in cutter an independent algorithm can be associated with other smoothing or filtering method has been summarized in this article.

Few of the existing FPGA and reconfigurable architectures along with techniques and methodologies used for increasing the performance of system for object-tracking applications are discussed. As mentioned, the reconfigurable hardware is distinguished due to ability to change the functionality post fabrication or even during operation. Reconfigurable custom architectures which includes reconfigurable processors aim to accelerate the software, and are therefore based on modified instruction set processors (ISP) architectures. Such type of architectures is therefore fixed. Reconfiguration in that case is confined to other subsections of the architectures, such as on reconfigurable fabric units similar to FPGAs. The reconfigurable pro-cessing fabrics (RPFs) in this case are categorized as fine-grained and coarse-grained. The fine-grained fabric manipulates data at bit-level; whereas, the coarse-grained fabrics operate a group of bits on reconfigurable fabric. As discussed, both have its own limitations and benefits. However, in term of performance, power and area utilization, coarse-grained RPFs are more optimized than its counterpart the fine-grained.

REFERENCES

1. E Cerro- prada, S M Charlwood, P B James-Roxby, “Designing Image Processing Applica-tions using Reconfigurable Computing”. Proc. Image processing and its applications. IEE- 1999.
2. Xilinx “ML605 Hardware User Guide”, UG534 (v1.8) Oct 2, 2013. www.xilinx.com
3. Díaz, Javier, et al. "FPGA-based real-time optical-flow system." Circuits and Systems for Video Technology, IEEE Transactions on 16.2 (2006): 274-279.
4. Kristensen, Fredrik, et al. "An embedded real-time surveillance system: Implementation and evaluation." Journal of Signal Processing Systems 52.1 (2008): 75-94.
5. Xilinx Soft Processors, Xilinx Inc., http://www.xilinx.com/microblaze.
6. Abderrahim Doumar, Kentaroh Katoh, Hideo Ito, “Fault Tolerant SoC Architecture Design for JPEG2000 using Partial Reconfigurability”. Proc. 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, Rome, Italy, Sept 26-28 (DFT 2007).
7. E.L. Horta, J.W. Lockwood, D.E. Taylor, D. Parlour, "Dynamic Hardware Plugins in an FPGA with Partial Run-time Reconfiguration", Design Automation Conference (DAC), June 2002.
8. Miguel Arias-Estrada, and Eduardo Rodríguez-Palacios, “An FPGA Co-processor for Real Time Visual Tracking”. Proc. LNCS 2438, pp. 710-719, 2002. Springer-Verlag Berlin Hei-delberg 2002.
9. Usman Ali, M.B. Malik and Khalid Munawar, “FPGA/Soft- Processor based real-time ob-ject tracking system”, Proceedings IEEE-2009.
10. D. J. Li, L Jiang T Isshiki, H. Kunieda, “New VLSI Array Processor Design for Image Window operation”, IEEE Transaction on circuits and systems-II: Analog and Digital Signal Processing. Vol. 46, No. 5, May 1999.
11. Gray, J and Kean T, “Configurable hardware: a new paradigm for computation”, Proc. Di-cennial CalTech Conference, 1989
12. Alper Yilmaz, O. Javed, and M. Shah, Object tracking: A survey, ACM Computing Sur-veys, Vol. 38, No. 4, 1-45, 2006

ISSN: 2583-2621

13. Comaniciu, Dorin, Visvanathan Ramesh, and Peter Meer. "Kernel-based object tracking." Pattern Analysis and Machine Intelligence, IEEE Transactions on 25.5 (2003): 564-577
14. SATO, K. AND AGGARWAL, J. 2004. Temporal spatio-velocity transform and its appli-cation to tracking and interaction. Comput. Vision Image Understand. 96, 2, 100-128.
15. SALARI, V. AND SETHI, I. K. 1990. Feature point correspondence in the presence of oc-clusion. IEEE Trans. Patt. Analy. Mach. Intell. 12, 1, 87-91.
16. Weng, Shiuh-Ku, Chung-Ming Kuo, and Shu-Kang Tu. "Video object tracking using adap-tive Kalman filter." Journal of Visual Communication and Image Representation 17, no. 6 (2006): 1190-1208.
17. SHI, J. AND TOMASI, C. 1994. Good features to track. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 593-600.
18. BERTALMIO, M., SAPIRO, G., AND RANDALL, G. 2000. Morphing active contours. IEEE Trans. Patt. Analy. Mach. Intell. 22, 7, 733-737
19. KANG, J., COHEN, I., AND MEDIONI, G. 2004. Object reacquisition using geometric invariant appearance model. In International Conference on Pattern Recongnition (ICPR). 759-762.
20. Chi, Jing, Shanshan Gao, Yunfeng Zhang, and Caiming Zhang. "A Region-based Expres-sion Tracking Algorithm for Spacetime Faces." Computers & Graphics (2015).
21. Liwicki, Stephan, Stefanos Zafeiriou, Georgios Tzimiropoulos, and Maja Pantic. "Fast and robust appearance-based tracking." In Automatic Face & Gesture Recognition and Work-shops (FG 2011), 2011 IEEE International Conference on, pp. 507-513. IEEE, 2011.
22. Jin, Jonghoon, et al. "Tracking with deep neural networks." Information Sciences and Sys-tems (CISS), 2013 47th Annual Conference on. IEEE, 2013
23. Zhu, Shan, and Kai-Kuang Ma. "A new diamond search algorithm for fast block-matching motion estimation." Image Processing, IEEE Transactions on 9.2 (2000): 287-290.
24. Mayer, Arnaldo, and Hayit Greenspan. "An adaptive mean-shift framework for MRI brain segmentation." Medical Imaging, IEEE Transactions on 28.8 (2009): 1238-1250.
25. Nie, Yao, and Kai-Kuang Ma. "Adaptive rood pattern search for fast block-matching motion estimation." Image Processing, IEEE Transactions on 11.12 (2002): 1442-1449.
26. Turk, Matthew, and Alex P. Pentland. "Face recognition using eigenfaces." Computer Vi-sion and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Con-ference on. IEEE, 1991.
27. Black, Michael J., and Allan D. Jepson. "Eigentracking: Robust matching and tracking of ar-ticulated objects using a view-based representation." Computer Vision—ECCV'96. Springer Berlin Heidelberg, 1996. 329-342.
28. Cheng, Yizong. "Mean shift, mode seeking, and clustering." Pattern Analysis and Machine Intelligence, IEEE Transactions on 17.8 (1995): 790-799.
29. Comaniciu, Dorin, and Peter Meer. "Mean shift: A robust approach toward feature space analysis." Pattern Analysis and Machine Intelligence, IEEE Transactions on 24.5 (2002): 603-619.
30. Comaniciu, Dorin, Visvanathan Ramesh, and Peter Meer. "Real-time tracking of non-rigid objects using mean shift." Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on. Vol. 2. IEEE, 2000.
31. Wang, Jue, Sunghyun Cho, and Shuaicheng Liu. "Simulating tracking shots from image se-quences." U.S. Patent No. 9,253,415. 2 Feb. 2016..
32. Emami, Ebrahim, and Mahmood Fathy. "Object tracking using improved CAMShift algo-rithm combined with motion segmentation." Machine Vision and Image Processing (MVIP), 2011 7th Iranian. IEEE, 2011.
33. Sutton, Michael A., Jean Jose Orteu, and Hubert Schreier. Image correlation for shape, mo-tion and deformation measurements: basic concepts, theory and applications. Springer Sci-ence & Business Media, 2009.
34. Wang, Naiyan, and Dit-Yan Yeung. "Learning a deep compact image representation for vis-ual tracking." Advances in Neural Information Processing Systems. 2013.
35. Bartlett, Marian Stewart, Javier R. Movellan, and Terrence J. Sejnowski. "Face recognition by independent component analysis." Neural Networks, IEEE Transactions on 13.6 (2002): 1450-1464.
36. Zhou, Huiyu, Yuan Yuan, and Chunmei Shi. "Object tracking using SIFT features and mean shift." Computer vision and image understanding 113.3 (2009): 345-352.
37. Tsai, Du-Ming, and Shia-Chih Lai. "Independent component analysis-based background subtraction for indoor surveillance." IEEE Transactions on image processing 18.1 (2009): 158-167.
38. Kaittan, Assel H., and Thamir R. Saeed. "Tracking of Video Objects Based on Kalman Fil-ter." Journal of University of Babylon 25.5 (2017): 1507-1518.
39. Blaber, J., B. Adair, and A. Antoniou. "Ncorr: open-source 2D digital image correlation matlab software." Experimental Mechanics 55.6 (2015): 1105-1122.
40. Nummiaro, Katja, Esther Koller-Meier, and Luc Van Gool. "An adaptive color-based particle filter." Image and vision computing 21.1 (2003): 99-110.
41. Rui, Yong, and Yunqiang Chen. "Better proposal distributions: Object tracking using un-scented particle filter." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Pro-ceedings of the 2001 IEEE Computer Society Conference on. Vol. 2. IEEE, 2001.
42. Kaew Tra Kul Pong, Pakorn, and Richard Bowden. "An improved adaptive background mixture model for real-time tracking with shadow detection." Video-based surveillance sys-tems. Springer US, 2002. 135-144.
43. Zivkovic, Zoran. "Improved adaptive Gaussian mixture model for background subtraction." Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. Vol. 2. IEEE, 2004.
44. Canny, John. "A computational approach to edge detection." Pattern Analysis and Machine Intelligence, IEEE Transactions on 6 (1986): 679-698.
45. Kim, Changick, and Jenq-Neng Hwang. "Fast and automatic video object segmentation and tracking for content-based applications." Circuits and Systems for Video Technology, IEEE Transactions on 12.2 (2002): 122-129.
46. Zhu, Zhigang, et al. "Panoramic virtual stereo vision of cooperative mobile robots for localiz-ing 3d moving objects." Omnidirectional Vision, 2000. Proceedings. IEEE Workshop on. IEEE, 2000.
47. Zhu, Shiping, et al. "A new cross-diamond search algorithm for fast block motion estima-tion." 2009 16th IEEE International Conference on Image Processing (ICIP). IEEE, 2009.
48. Xilinx, Inc. XC6200 Field Programmable Gate Arrays Product Description, Xilinx, Inc., San Jose, 1997.
49. Atmel Corp. AT40K Series FPGA Interactive Architecture Guide. Atmel Corp., San Jose, 1999
50. Celoxica, RC2000 Development and evaluation board data sheet,version 1.1, 2004
51. Xilinx, Inc., PowerPC 405 Processor Block Reference Guide, October 2003
52. Altera Corp., Stratix II Device Handbook, February 2004
53. Altera Corp., Excalibur Device Overview, May 2002
54. Hauser, John R., and John Wawrzynek. "Garp: A MIPS processor with a reconfigurable coprocessor." Field-Programmable Custom Computing Machines, 1997. Proceedings., The 5th Annual IEEE Symposium on. IEEE, 1997.
55. Singh, H., Lee, M.-H., Lu, G., Kurdahi, F., Bagherzadeh, N., and Chaves, E.: MorphoSys: an integrated reconfigurable system for dataparallel and compute intensive applications', IEEE Trans. Comput., 2000, 49, (5), pp. 465–481
56. T. Miyamori and K. Olukotun, "REMARC: Reconfigurable Multimedia Array Coproces-sor", Proceedings ACM International Symposium on Field-Programmable Gate Arrays, 1998.

ISSN: 2583-2621

57. B. Mei, S. Vernalde, D. Verkest, H. De Man and R. Lauwereins, “ADRES: An Architecture with Tightly Coupled VLIW Processor and Coarse-Grained Reconfigurable Matrix”, Pro-ceedings Field-Programmable Logic and Applications, 2003.
58. Goldstein, Seth Copen, et al. "PipeRench: A reconfigurable architecture and compiler." Computer 33.4 (2000): 70-77.
59. Ebeling, Carl, Darren C. Cronquist, and Paul Franklin. "RaPiD—Reconfigurable pipelined datapath." Field-programmable logic smart applications, new paradigms and compilers. Springer Berlin Heidelberg, 1996. 126-135.
60. Happe, Markus, Enno Lübbers, and Marco Platzner. "A self-adaptive heterogeneous multi-core architecture for embedded real-time video object tracking." Journal of real-time image processing 8, no. 1 (2013): 95-110.
61. E. Caspi, M. Chu, R. Huang, J. Yeh, J. Wawrzynek and A. DeHon, “Stream computations organized for reconfigurable execution (SCORE)”, Proceedings Field Programmable Logic and Applications, 2000.
62. Wang, Ying, Xuegong Zhou, Lingli Wang, Jian Yan, Wayne Luk, Chenglian Peng, and Jia-rong Tong. "Spread: A streaming-based partially reconfigurable architecture and program-ming model." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 21, no. 12 (2013): 2179-2192.
63. P. N. Green and M. D. Edwards, “Object oriented development method for “reconfigurable embedded systems”, IEE Proceedings Computers and Digital Techniques, Vol. 147, No. 3, pp 153-158, May 2000.
64. K. Hazelwood, W. B. Ligon, G. Monn, N. Pothen, R. Sass, D. Stanzione and K. D. Under-wood, “Creating Applications in RCADE”, IEEE Aerospace Conference, 1999.