A FPGA based fast runtime reconfigurable real-time Multi-Object-Tracker

Matthias Rümmele-Werner, Thomas Perschke
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB
Ettlingen, Germany

Lars Braun, Michael Hübner, Jürgen Becker
Institute for Information Processing Technologies
Karlsruhe Institute of Technology
Karlsruhe, Germany

Abstract— **This paper presents a real-time Multi-Object-Tracker implemented on a Field Programmable Gate Array (FPGA). This system is able to track three objects simultaneously using different algorithms to get the best result. Each algorithm has its own field of application and the user can decide which algorithm is used for individual objects. Using the dynamic, partial reconfiguration capability of Xilinx FPGAs, the algorithms can be exchanged during runtime without interrupting the object-tracking. To obtain a self reconfigurable system the Internal Configuration Access Port (ICAP) is used. In this application the needed time to exchange the algorithms has to be as short as possible. In this paper we present a design to achieve the theoretical maximum throughput of the ICAP of 400 MB/s.**

I. Introduction And Related Work

Object tracking becomes important in industry and security applications. Especially monitoring activities requires different efficient algorithms for object tracking. For processing image sequences in real-time we need high-performance hardware such as digital signal processors (DSP), general purpose graphic processor units (GPGPU) or Field Programmable Gate Arrays (FPGAs).

During the last years, performance and flexibility of FPGAs have increased strongly. It is possible to use FPGAs for complex systems and applications such as image processing. For this reason FPGAs are also a good option for object tracking. Additionally, some FPGAs support dynamic, partial reconfiguration (DPR) to provide higher flexibility. With this mechanism it is possible to reconfigure a part of the FPGA during runtime. The other part of the FPGA is unaffected by this process and continues to run. A lot of research has been done on the topic of partial, dynamic reconfiguration. For example, [1] describes how to build a new ICAP-controller to reach maximum throughput for reconfiguration.

Several implementations of object tracking for FPGAs exists. In [4] is demonstrated how to build up a soft-processor based real-time object tracking system. In this paper they used the Xilinx 32bit Microblaze soft processor to implement the tracking algorithm. The rest of the FPGA is used for the frame grabber and the visualization of the video stream. [5] describes a multi-object-tracking architecture for FPGAs or ASICs based on image segmentation. These systems have some restrictions. The main disadvantage is the restriction of the systems to one algorithm. If the constraints are changing, the settings of an algorithm has to be changed. In the worst case the algorithm gets completely improper. To avoid this problems the possibility to be responsive of changing constraints is needed to get the best results.

In this paper a novel approach is presented how to build up a FPGA based real-time multi-object-tracker. The system combines different hardware accelerated tracking algorithms on one platform using a Virtex-4 FPGA from Xilinx. It is possible to track multiple objects with different algorithms at the same time. It is also possible to track one object with different algorithms simultaneously. A structure which provides an easy exchange of algorithms with the opportunity to add easily new additional algorithms is presented. For additional functionality DPR is used to exchange tracking algorithms during runtime. For exchanging the algorithms as fast as possible a new ICAP-controller is introduced for reaching the maximum throughput.

This paper is organized in the following manner: In Section II the system architecture is described. In II-A the tracking part of the system will be explained. In II-B the reconfiguration-control-system is described in detail and in II-C the new ICAP controller is characterized. In Section III the software running on the Host-PC is explained. In Section IV an experimental evaluation of the system is done. Conclusion and outlook are given in section V.

II. The System Architecture

The system can be divided in three main components which are the Host-PC, the FPGA and the camera. In this paper we will focus on the FPGA part but for a better understanding the whole system will be described shortly in this section.

The camera, which we use as data source, has a resolution of 384x286 pixels and provides a pixel stream with a frame rate of 25 Hz. The camera is directly connected to the FPGA. For caching the input data and changing the clock domain several FIFOs are used. The system provides three independent tracking paths. A tracking-path requires three components: a input-FIFO, a control-module and a tracker. As different tracking algorithms have different throughput rates, one input-FIFO per tracking-path is needed. An additional input-FIFO is used to provide the camera data for visualization on the Host-PC. The control-module ensures that the tracking starts at the right point of time and the tracker sends the result to the output multiplexer. Finally the results of the trackers are stored together with the frame data in the output-FIFO from where the Host-PC can collect the data.

To control and reconfigure the FPGA system an instance for the communication with the Host-PC is needed. As a Virtex 4 FX140 is used, the PowerPC 405 is chosen because it already exists as hard IP-core on this FPGA. To communicate with the Host-PC, the system uses a RS232 interface. Additionally, some modules for reconfiguration are needed. The physical reconfiguration is done by the ICAP, included in the ICAP-controller. The ICAP is a internal configuration interface of the FPGA which can be used for self configuration. The start of the reconfiguration is triggered from the Host-PC via the PowerPC. The partial bitfiles which are needed for reconfiguration are stored in the onboard DDR2 memory. The ICAP has direct access to the DDR2 memory controller.

The Host-PC has two tasks. The first task is the controlling of the whole tracking-system and to provide the interface for the user. Additionally, the Host-PC adds the results of the tracking-algorithms to the images and visualizes them on the monitor.

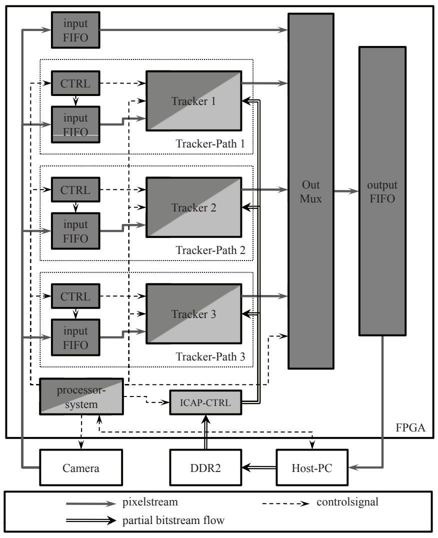

Fig. 1. System architecture of the multi-object-tracker

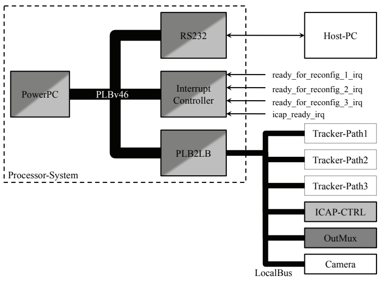

Fig. 2. Modules and connection of the processor-system

The FPGA design can be divided in two major parts: the tracking-system and the reconfiguration-control-system. A detailed description of these two parts is given bellow:

A. The tracking-system

In Fig. 1, the dark grey arrows and modules mark the dataflow of the camera. The incoming camera data are deserialized with the camera interface. This data are cached in the four input FIFOs which are used for changing the clock domains from the 7.5 MHz camera clock to the 100 MHz system clock. Three of this input FIFOs belong to the three independent tracker-paths. The output of the fourth FIFO is directly mapped to the output multiplexer (OutMux). This FIFO provides the images which are visualized on the Host-PC. After the system start all trackers are deactivated by default. The user has to select an object in the displayed camera images using the GUI (Graphical User Interface). After this action the object will be tracked. The control-module CTRL ensures that the tracking starts at the beginning of a new frame. If a tracking-path is not used, the control-module sets all other modules in this path into a reset state. The tracker-modules of all three paths contain by default a correlation algorithm. During runtime it is possible to exchange the correlation algorithm with two other algorithms: a hotspot or a centroid algorithm. Due to the page limitation only a short description of this algorithms is given below.

- Correlation-Tracker: The algorithm aligns a template T with the current frame with intensity values $I\,(x, y)$ at the point (u, v) and calculates the normalized cross correlation κ given by

$$\kappa = \frac{\sum_{x,y}\left(I\,(x,y) - \bar{I}_{u,v}\right)\left(T\,(x-u, y-v) - \bar{T}\right)}{\sqrt{\sum_{x,y}\left(I\,(x,y) - \bar{I}_{u,v}\right)^2 \sum_{x,y}\left(T\,(x-u, y-v) - \bar{T}\right)^2}}$$

  where the bar indicates the mean of the values. From these values the position of the maximum value is chosen as the new coordinate of the object. This tracker is used for small to large objects with texture. In this case the Correlation-Tracker is more robust than the Centroid-Tracker in respect of perturbations.
- Hotspot-Tracker: The brightest spot is searched with a robust algorithm to avoid wrong results due to pixel defects. The intensity values of a 3×3 kernel around a pixel are sorted and the median M and the quartiles $Q_{1,3}$ are used to calculate a new intensity value $\hat{I}$ given by

$$\hat{I} = 1/4\,(Q_1 + 2M + Q_3)$$

  The coordinate of the maximum value of $\hat{I}$ are used as the new object coordinate. The algorithm is typically used for bright small objects.
- Centroid-Tracker: The Centroid-Tracker segments the image by calculating the mean and the variance of the intensity values. These are used to define a threshold. All pixels with intensity values above the threshold are defined to belong to the object. The new position is calculated as the weighted mean

$$x^{new}(y^{new}) = \sum_{x,y \in \text{object}} x(y) w\,(x, y) \,/ \sum_{x,y \in \text{object}} w\,(x, y)$$

  of the position of this pixels where the weight $w\,(x, y)$ is either the intensity value $I\,(x, y)$ or 1. This algorithm is used for small to large objects without texture as the calculated object position is much more stable than positions given by the Hotspot-Tracker or the Correlation-Tracker.

The results of each algorithm is put out at the end of each frame. The exchange of the tracker is done via DPR during runtime and will be discussed in detail below. Each tracker sends the results to the output multiplexer (OutMux in Fig. 1). For every frame which runs through a active tracking-path we get one result. That means if all tracking-paths are activated we get three results per frame. The easiest way to transmit the result to the Host-PC is to append the results to the frames. Therefore, the OutMux needs same additional logic to recognize the frame end. At this point, the result data are added to the end of a frame. From the OutMux-module the data are written to the output FIFO. This FIFO is a 16 bit to 64 bit FIFO as the Host-PC collects the data in 64 bit datawords via DMA-transfer.

B. The reconfiguration-control-system

In this section the system part responsible for DPR is explained in detail. The light gray components in Fig. 1 belong to the reconfiguration-control-system. The most important component of this module is the processor-system. This part of the system was implemented with Xilinx EDK tool. It is responsible for the communication with the Host-PC and for the correct execution of the DPR. The processor-system is divided in four separate modules which are shown in Fig. 2. The PowerPC is the controlling instance of the system and is connected via the PLB bus to the RS232 IP-core, the interrupt controller and the PLB2LB Bridge. The PowerPC runs at a frequency of 200 MHz and the tracker-control-application running on the PowerPC communicates over the RS232-module with the Host-PC. The tracker settings are given by the user and are transmitted via RS232 to the tracker-control-application. The application forwards

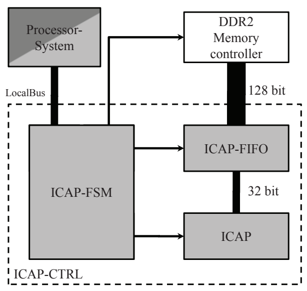

Fig. 3. ICAP controller

the settings to the corresponding modules. If another tracking-algorithm is wanted, the user has to set the desired option with the GUI. After this, the Host-PC sends a request for reconfiguration to the tracker-control-application. The reconfiguration-process is split up in two parts. The first part is the physical reconfiguration which is done by the ICAP-controller and the second part is to transmit the new tracker settings. The reconfiguration process starts when the OutMux-module triggers the *ready_for_reconfig_1_irq*-signal (Fig. 2) which is connected with the interrupt controller of the processor-system. This ensures that the reconfiguration process starts at the right point in time. This interrupt signal exists for every tracking-path. It is triggered if the end of a frame is reached and the tracking results are transmitted to the output FIFO. After this, the tracker-control-application starts the reconfiguration process using the ICAP-controller. When the ICAP-controller has finished the reconfiguration, the *icap_ready_irq*-signal is triggered to inform the tracker-control-application that the reconfiguration has finished and the new tracker-algorithm is available. The same interrupt signal is also connected with the Host-PC. If the Host-PC recognizes this interrupt the second part of the reconfiguration process starts. The new tracker settings will be sent to the FPGA and the object tracking will be started.

C. The ICAP-controller

Due to the big size of the partial bitstreams which are used in this design (table II) the reconfiguration time is very important. For real-time tracking, the reconfiguration time has to be minimized. In an ideal situation we can reconfigure the system between two frames. The timespan for reconfiguration is 1.9 ms. If the reconfiguration process exceeds the 1.9 ms the algorithm can not deliver any results for frames which are in this reconfiguration time because it is not possible to start the tracking in the middle of a frame. In [1], [2] and [3] the XPS_HWICAP is investigated in detail. In [2] different ICAP implementations are examined focused on reconfiguration time. The fastest design reaches an average speed of 332.1 MB/s with the disadvantage that the bitfiles are directly stored in BRAMs on the FPGA. The ICAP-design in [3] stores the bitfiles in the DDR2-memory and reaches 400.0 MB/s. Other ICAP-designs connected via the PLB bus with the memory-controller are much slower and do not reach 100 MB/s. Concerning this results it is obvious that the standard XPS_HWICAP does not fulfill our requirements to reach such short reconfiguration times with the big bitfiles (table II).

In [1] is described how to build an ICAP-controller to reach maximum throughput, but this ICAP-controller is still connected via the PLB bus with the memory controller which is the bottleneck. In [3] the ICAP-controller is directly connected with the standard memory-controller. The communication is still done via the PLB-bus. In our new approach the ICAP-controller is connected directly with memory-controller without using the PLB-bus and the standard memory-controller. It is illustrated in fig. 3. This ICAP-controller is made up of three modules: the ICAP-FSM, the ICAP-FIFO and the ICAP itself. The partial bitfiles which are needed for reconfiguration are slightly modified before they are stored in the onboard DDR2 memory. The memory controller, which is directly connected to the ICAP-controller, works at a frequency of 200 MHz and reads the data in 16 Byte words. They are stored in the ICAP-FIFO which is used for caching the data. The ICAP runs with the maximum frequency of 100 MHz but has only a 4 Byte data input. Due to this reasons it is avoided that the ICAP-FIFO runs empty during the reconfiguration process. The ICAP reaches the maximum throughput of 400 MB/s.

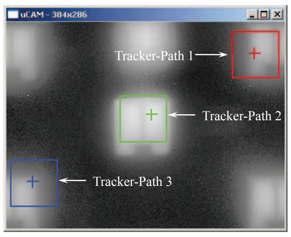

Fig. 4. Screenshot of the visualization window

III. User Interface

In this section the user interface implemented as C++ Software on a x86 Windows XP system (Host-PC) will be described. The software contains four different threads, which are initialized during the startup procedure. The first step of the software is to allocate the shared memory which is used by three threads: two DMA-reader threads and the thread for visualization of the camera data. After this step the FPGA will be configured. Now the DMA-channels are initialized. As we have two DMA-Channels we have two DMA-threads for reading or writing the data from or to the FPGA. This is followed by storing the partial bitfiles into the DDR2-memory. At this point the startup of the FPGA-system has finished. Therefore, the DMA-reader threads, the visualization-thread and the user-interface-GUI-thread are started. The DMA-reader threads collect the image data from the FPGA (output-FIFO) and store them into the shared memory. From there the visualization-thread reads the data and visualize them in the corresponding window (Fig. 4). Additionally the result data from the tracker algorithms are processed and the location of the tracked object is overlayed into the image using colored crosses.

To interact with the user a second window is needed. With this user-interface-GUI it is possible to adapt the settings of the individual tracker algorithms during runtime. The user has the possibility to choose between three different algorithms and can start the tracking by clicking directly on the object of interest in the visualization window. The possibility of stopping and restarting the tracking process is given. The exchange of the algorithms using DPR is also handled with the user-interface-GUI.

IV. Experimental Results

To evaluate the performance of this multi-object-tracker, some experiments have been done. Due to the fact that the reconfiguration time is the most important factor in this system, the focus is first set on this property. The system is implemented on a Xilinx Virtex4 FX-140 FPGA on a PCI mezzanine card from Alpha-Data (ADM-XRC-4FX). This card is combined with the ADM-PMC carrier. So the card can be used with a standard PC using the PCI-interface. The processor-system and the memory-controller run at a frequency of 200 MHz. The rest of the FPGA-system runs at a frequency of 100

TABLE I
THE HARDWARE UTILIZATION OF THE VIRTEX 4FX 140

	Slices	4 input LUTs	Slice FFs	RAMB16	DSP48
Static part	8333 - 13.2%	9379 - 7.4%	22481 - 17.8%	61 - 11.1%	-
Correlation Tracker	5244 - 8%	6578 - 5%	5946 - 4%	30 - 5%	55 - 28%
Hotspot Tracker	1788 - 2%	2226 - 1%	1566 - 1%	-	-
Centroid Tracker	1327 - 2%	1895 - 1%	1813 - 1%	10 - 1%	13 - 6%

TABLE II
EVALUATION OF THE RECONFIGURATION TIME OF THE ICAP-CONTROLLER ($f_{sys} = 100$ MHZ) FOR TRACKER-PATH 3

Bitfile	Bitfilesize	Num. of Datawords (n_{dw})	Theoretical reconfig. time (t_{tr})	Real reconfig. time (t_{rr})	Reconfig. speed
Correlation Tracker Slot 3	691.584 kB	172896	1.72896 ms	1.72896 ms	400.00 MB/s
Hotspot Tracker Slot 3	538.528 kB	134632	1.34632 ms	1.34632 ms	400.00 MB/s
Centroid Tracker Slot 3	574.080 kB	143520	1.4352 ms	1.4352 ms	400.00 MB/s

MHz, except the camera interface which uses the 30 MHz camera clock.

A. Reconfiguration time

To measure the reconfiguration time, a hardware counter was implemented in the ICAP-control-module. This counter counts the clock cycles which are required for a reconfiguration process. Based on the number of datawords n_{dw} of a bitfile, the minimum number of needed clock cycles n_{cc} is known $n_{dw} = n_{cc}$. The frequency of the ICAP-controller is given by f_{sys} =100 MHz. So the minimum theoretical reconfiguration time t_{tr} can be calculated in the following manner:

$$t_{tr} = \frac{n_{dw}}{f_{sys}}$$

The result of the hardware counter, is the measured number of clock cycles n_{mcc} the system needs for reconfiguration. With this value it is possible to calculate the real reconfiguration time t_{rr}:

$$t_{rr} = \frac{n_{mcc}}{f_{sys}}$$

If t_{tr} is smaller than t_{rr} the ICAP-FIFO runs empty and the reconfiguration process is paused. This means the memory-controller is to slow and the maximum throughput of the ICAP can not be reached. For the multi-object-tracker some experimental measurements have been done. Every partial bitfile was used ten times for reconfiguration. The measurement results for tracker-path 3 are listed in table II. In our design the ICAP-FIFO never runs empty and though we reach the maximum throughput of 400MB/s. In this case t_{tr} is equal to t_{rr}.

The bitfile with the biggest size is the correlation-tracker algorithm of tracker-path 3. The reconfiguration time of this bitfile is 1.72896 ms. The timespan between two frames is about 1.9 ms. With this type of ICAP-controller it is possible to exchange algorithm between two frames without stopping the tracking process.

B. Hardware utilization

In the case of the multi-object-tracker DPR reduces the needed chip size significant. The hardware utilization of the static part and the algorithms is shown in table I. Each of the three partial reconfiguration regions has a size of 1664 CLBs (Complex Logic Block). This size is needed to place the correlation algorithm. Although the other algorithms do not need as much hardware as the correlation tracker, it will not be possible to place all algorithms on one FPGA during the same time caused by the number of used DSPs. DPR provides a good method to decrease chip size and to run the multi-object-tracker on one FPGA.

V. CONCLUSION AND OUTLOOK

In this paper the implementation of a fast runtime reconfigurable real-time multi-object-tracker is described. We implemented a new ICAP-controller which reaches the maximum throughput of 400 MB/s. With this high-speed reconfiguration technique it is possible to exchange the tracking algorithms between two frames and the system provides the opportunity to exchange tracking algorithms during runtime without stopping the tracking process. As result the tracked object does not get lost due to exchanging the algorithms. The next step to reach an autonomous system is to develop a logic which exchanges the algorithms by itself.

To reduce complexity of the implementation of reconfigurable systems, we want to develop a framework for algorithms with a corresponding development environment for an easier handling of the development of reconfigurable systems.

REFERENCES

[1] C. Claus, B. Zhang, W. Stechele, L. Braun, M. Hübner, J. Becker, *A multi-platform controller allowing for maximum Dynamic Partial Reconfiguration throughput*, Proc. of the International Conference on Field Programmable Logic and Applications, pp. 535-539, 2008.

[2] Liu Ming, W. Kuehn, Lu Zhonghai, A. Jantsch, *Run-time Partial Reconfiguration speed investigation and architectural design space exploration*, Proc. of the International Conference on Field Programmable Logic and Applications, pp. 498-502, 2009.

[3] C. Claus, R. Ahmed, F. Altenried, W. Stechele, *Towards rapid dynamic partial reconfigurationin video-based driver assistance systems*, 6th Internation Symposium of Reconfigurable Computing: Architectures, Tools and Applications, pp. 55-67, 2010

[4] A. Usman, M. B. Malik, K. Munawar, *FPGA/soft-processor based real-time object tracking system*, Proc. of the 5th Southern Conference on Programmable Logic, pp. 33-37, 2009.

[5] K. Yamaoka, T. Morimoto, H. Adachi, K. Awane, T. Koide, H.J. Mattausch, *Multi-object tracking VLSI architecture using image-scan based region growing and featured matching*, Proc. of the International Symposium on Circuits and Systems, ISCAS 2006.

[6] Alpha Data, http://www.alpha-data.com

[7] Xilinx, *Virtex-4 FPGA user guide*, ug070 v2.6, 2008.

[8] Xilinx, *Virtex-4 FPGA configuration user guide*, ug071 v1.11, 2009.

[9] Xilinx, *Early access partial reconfiguration user guide*, ug208 v1.2, 2008.