A perfomance comparison study between synchronous and asynchronous FPGA for spike based Systems.

Under the AER synthetic generation

Rafael Paz-Vicente, Elena Cerezuela-Escudero, Manuel Dominguez-Morales, Angel Jimenez-Fernandez, Alejandro Linares-Barranco and Gabriel Jimenez-Moreno

Arquitectura y Tecnología de Computadores. Universidad de Sevilla.
Av. Reina Mercedes s/n, 41012-Sevilla, SPAIN
rpaz@atc.us.es

Abstract— **Neuromorphic engineering tries to mimic biology in information processing. Address-Event Representation (AER) is a neuromorphic communication protocol for spiking neurons between different layers. AER bio-inspired image sensors are called "retina". This kind of sensors measurees visual information not based on frames from real life and generates corresponding events. In this paper we provide an alternative, based on cheap FPGA, to these image sensors that takes images provided by an analog video source (video composite signal), digitalizes it and generates AER streams for testing purposes. This design was initially developed for Xilinx Spartan FPGA. In this paper we present a comparison study between synthesis of this design for Xilinx Spartan, Virtex FPGA and Achronix asynchronous FPGA, measuring the maximum performance reached in each case.**

Keywords-Address-Event-Representation, AER synthetic generation, visual processing, frame-free vision, FPGA, VHDL, asynchronous logic.

I. INTRODUCTION

In these days, FPGA has been popularized in industrial, educational and research environment due to its increasing capability and speed, for a reasonable prize. They are very useful particularly in research environment because they allows fast prototyping for implementation of digital designs, that can be changed without modification of hardware, just changing logic design programmed on it. Developers can modify and test their high speed design, without having to develop ASIC chips, that requires long times of manufacturing, high prizes, and many previous simulation to ensure that the design works properly. May be one of the most popular families of FPGA from some years are those manufactured by Xilinx, that continuously develops new faster and bigger FPGA.

On the other hand, new companies have emerged offering new and innovative solutions, trying to present attractive alternatives to existing ones, or trying to solve weak points on current technologies. One of these recently created companies is Achronix Semiconductors. It presents a new type of FPGA that it promises that can reach top speeds of about 1.5 Ghz, or three to four times speed improvement comparing to existing top performance FPGAs. Achronix Speedster FPGA was the first family of FPGAs provided by Achronix, and this is the one we have used in this study. Recently they released a new family, Speedster22i, using 22nm technology, and it has signed a contract with Intel for manufacturing hybrid chips.

Achronix FPGA is very different from traditional FPGA because internally their logic is completely asynchronous. Removing global clock allows avoiding clock propagation problems, and taking the concept of pipeline to the limit, and in theory, a well optimized design can reach clocks speeds of 1.5 GHz. Anyway, we have to take in consideration, that this high speed clock does not means that any process will take one clock cycle to finish, but that the pipeline can run so fast, but use to have many stages.

Although internally Achronix Speedster works in an asynchronous way, provided designs has to be clocked. Speedster input ports are synchronous and input and output are latched. Synthesis software takes synchronous designs and maps it to fit on internal asynchronous logic to work properly.

In this paper we present a design developed initially for Xilinx Spartan II 200 FPGA that lately we have re-synthetized for Xilinx Virtex family and also for Achronix Speedster. We compare synthesis time spent by synthesis software, for these three cases. We also compare maximum clock frequencies that can be reached by each design, and resources used by each case.

Proposed design was developed for being used as a test tool in developing AER systems for real time vision processing, as a source of spikes. AER bus is a neuroinspired bus used in spiking systems for video, audio or in general, sensorial information processing.

II. AER BUS DESCRIPTION

Biology provides efficient solutions for several problems. Trying to mimic biology, bio-inspired and neuro-inspired systems have been extended in the last years. High degree of parallelism and scalability makes this emerging computing

technology especially interesting for real time vision processing.

One of the neuro-inspired models which mimic the neuron layers in the brain are the spiking models. These models can use the AER protocol to communicate spikes between different layers. There is a growing community of AER system developers. We can find several sensors (cochleas, retinas…)[1][2][3], processing layers (filters, convolutions, WTA…)[4][5][6][7], robotic controlling (Central Pattern Generators, Eddie…)[8][9], computer interface for system monitoring and sequencing (PCI-AER, USB-AER, USBAERmini2…)[10][11], and computer software for real-time processing (jAER, MATLAB interface)[12][13]. The goal of this community is to build large multichip and multi-layer hierarchically structured systems, for real-time massively-parallel spike processing. One of the most extended fields is the vision processing using an AER chain for objects recognition, tracking, etc…[14][15][16] Figure 1 shows the biggest AER chain developed for vision processing under the EU CAVIAR project, composed by a temporal contrast retina, 2 convolution chips, 1 WTA filter chip and one learning chip, connected using mapper boards and inserting AER monitors for debugging purpose. Figure 7 shows synthetic retina used as first element in AER chains.

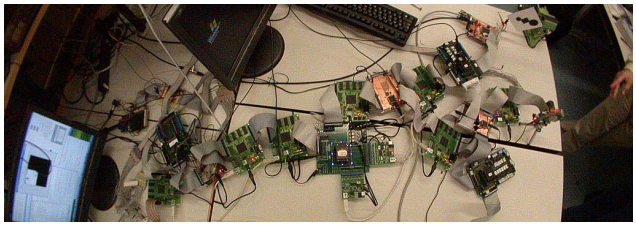


Figure 1. An AER chain for vision processing.

Spiking neurons codifies the information in a stream of pulses (spikes), whose frequency (or spike rate) is proportional to the neuron's excitation, following a Pulse Frequency Modulation (PFM). AER was proposed by Mead lab in 1991[17], as we have said before, for communicating neuromorphic chips with spikes. If we have several layers with hundred or thousand neurons, it becomes impossible to use a point to multi-point connection between every layer. AER tries to avoid this problem. The idea of the AER is to have only one common bus, the AER bus, giving a digital code address to every neuron. Every time a neuron fires a spike, it goes to an arbiter that manages collisions [15], and to an encoder, who writes the address in the AER bus (figure 2). Additional lines of request (REQ) and acknowledge (ACK) are required to communicate the address using a 4-step asynchronous hand-shake protocol. In the receiver, neurons will be listening the bus, looking for the spikes sent to them. Neurons are virtually connected by streams of spikes.

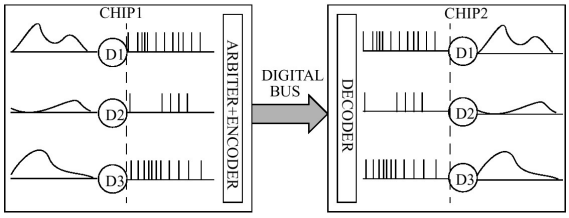


Figure 2. Spiking neurons communication between two layers using AER.

First layer in an AER image processing chain is an AER image sensor, also called retina that captures images from real life and generates AER event describing them. There are several ASIC AER retinas [1][2] that can be used for this purpose. Some of these retinas provide events describing pixel luminosity, but others send only luminosity changes information. This last group is called derivative retinas. Independently of the kind of retina we are using, both of them are implement in ASIC CMOS chips, so high knowledge of chip designing is required, many different polarization voltages are used, they have very complex biases to adjust, and due to expensive of custom chip prototyping, this technology is out of the possibilities of some developing groups, or at least the access to this devices can be very limited.

To popularize AER bus and neuro-inspired spiking systems, cheap image sensor that can be widely available, may be very interesting because that makes possible the development of AER filters, testing and debugging them. Furthermore, these sensors make this technology available for a more extensive community.

In this paper we present a comparison study of three implementations of a synthetic retina for FPGA using frame-based video source with a frame to AER conversion. Proposed synthetic retina try to provide this AER developing tool, working in a very similar way of real AER retinas, but at a lower cost. For that, most of the implementation of the synthetic retina consists in a digital circuit that is implemented in VHDL to be synthesized to fit on a CPLD or FPGA programmable digital device.

Image sensor consists in a cheap CMOS or CCD camera that output video composite signal. Alternatively, signal provided by a Video Recorder, DVD player, or computer output can be used. These video composite signals are captured using an ADC converter, and digitalized information is processed by digital logic to convert this video signal to AER events.

III. SYNCHRONOUS FPGA

In this section, we describe synchronous FPGA, focused in Xilinx FPGA, using for that Spartan and Virtex families. With superior FPGA and CPLD products, Xilinx claim to be the leader in digital programmable logic devices.

In general a FPGA is an integrated circuit designed to be configured by the designer, so it can be programmed. It contains programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", as desired by the designer.

At each configurable logic block (CLB) we can find usually a register (of one or more bits), driven by a clock signal, and some combinational logic, as multiplexors, adders, logic gates, etc. These components allow the designer to implement fully combinational blocks, or with the use of the mentioned register, to get the implementation of any kind of synchronous circuit, as several bit registers (combining several CLBs), shift registers, finite states machines (FSM), etc.

FPGA manufacturer (Xilinx in our case), usually provides design guidelines to help the designer to make implementation that best fit with specific FPGA internal architecture, getting better and faster designs.

Every CLB can be interconnected to others, using programmable interconnection matrixes. Synthesis tool, first translates described circuit by designer using a hardware description language (HDL, as for example VHDL or verilog) to a circuit. Then it chooses which parts of the circuit fits on each CLB to make physical placement, attending which parts of it has to be connected with which others, choosing properly their proximity and trying to minimize propagation delays in critic parts due to lines lengths by choosing the closest CLBs. It also determines the way to configure interconnection elements to properly interconnect different parts.

Usually, synchronous FPGA, in addition to generic I/O pins, has specific clock pins to get external clock signal. These clock pins, instead of being internally connected to generic I/O buffer, they are connected to fast buffers, that are connected to internal special lines that homogeneously distributes clock signal internally to reach every block in FPGA minimizing propagation delays and clock skew problems. Optionally, with specific PLL or DLL (phase locked loop or delay looked loop), this clock signal can be scaled or multiplied inside FPGA to get faster clock signals.

As described in next section, this clock distribution can be a limitation factor when trying to design high speed circuits. Usually, combinational path complexity is most important limitation factor, but even for very well optimized design, where combinational complexity is minimized, and divided in different stages with a fine grained pipeline, that allows speed up clock signal; clock propagation problem still limits maximum reachable clock frequency.

IV. ASYNCHRONOUS FPGA

In this section we explain asynchronous FPGA basics, based on Achronix Speedster FPGA, first designer of an asynchronous FPGA.

Achronix Semiconductor is a privately held fabless corporation based in Santa Clara, California. Achronix claims that he builds the world's fastest field programmable gate arrays (FPGAs) which use a unique patented circuit technology, providing 1.5 GHz throughput, a significant performance advantage over traditional FPGA technology.

The design of this FPGA takes a completely new and different approach when comparing with traditional FPGA design.

Traditional FPGA design is based in synchronous (clocked) reconfigurable pieces, named CLBs (Complex Logic Blocks). Clock signal is usually provided by selected pins, and internally it has to be distributed through all FPGA logic, avoiding skew and delay problems.

Achronix FPGA design is radically different. Internally, it is implemented by asynchronous logic, eliminating clock propagation problems. Externally, its behavior is similar to synchronous FPGAs, and synthesis tool provided by manufacturer synthetizes most synchronous designs. It's internal design is based in picoPIPEs. Each picoPIPE is formed by an array of RLBs (Reconfigurable Logic Blocks) connected by reconfigurable fabric. Each picoPIPE is surrounded by conventional synchronous I/O frames.

Traditional synchronous FPGA use a global clock that has to be propagated globally through the whole FPGA. Clock propagation is a limitating factor in maximum clock speed in designs. Even in pipelined design, where logic propagation delays effects can be minimized dividing global delay in several pipelines stages, the need to distribute clock signal over all FPGA fabric, becomes a limitation when working with high speed clock frequencies.

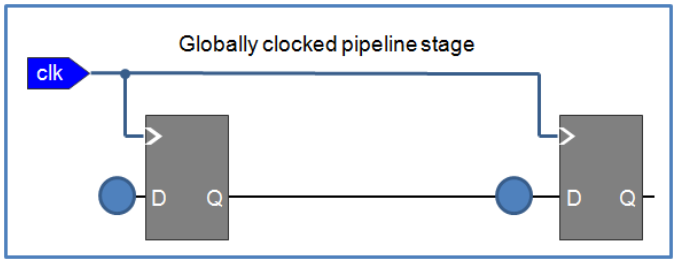


Figure 3. Globally clocked pipeline stage in synchronous FPGA.

Achronix FPGA eliminates clock signal. Data are transferred asynchronously from one stage to the next one. To transfer one data bit, internally Achronix Speedster FPGA uses two signals to transfer just one bit. Data is named token, so with using this two lines, it is possible to signal when there are a data or not been transferred, and in the case that there are some data, it's logic level.

When a token is received in one stage, an acknowledge signal is sent back to previous stage, so next data can be transferred. Using this method, there are implicitly defined a pipeline between stages, allowing to reach maximum clock speed.

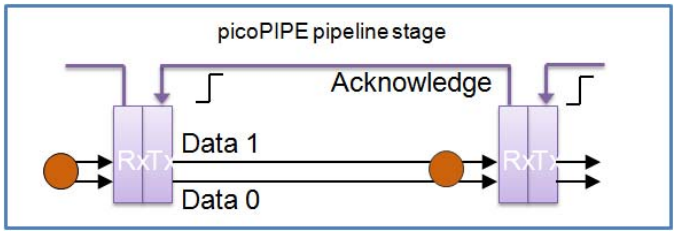
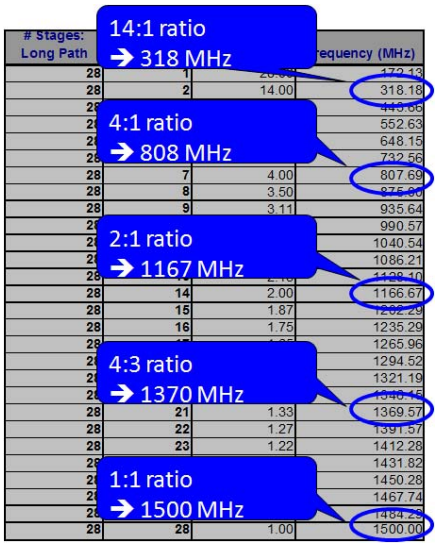


Figure 4. Asynchronous Achronix Speedster FPGA pipeline.

In this kind of FPGA, clock signal defined in VHDL or verilog code is used to determine chronologic succession of

events, so asynchronous design behaviour were similar to described synchronous design, but without using a clock in real implementation. When working with this kind of FPGA, limitation factor for clock speed is not large combinational path, like in standard FPGAs, but that combinational path is not balanced, combining larges and smalls combinational path. In this case, small combinational paths constitute a limitation for pipeline. Synthesis tool allows to artificially introduces additional stages in small combinational path to equalize length.

Maximum clock speed is determined by relation between largest and smallest combinational path, like shown in figure 5.



# Stages: Long Path			requency (MHz)
28	1		179.13
28	2	14.00	318.18
28			443.66
28			552.63
28			648.15
28			732.56
28	7	4.00	807.69
28	8	3.50	875.86
28	9	3.11	935.64
28			990.57
28			1040.54
28			1086.21
28			1128.10
28	14	2.00	1166.67
28	15	1.87	1202.29
28	16	1.75	1235.29
28			1265.96
28			1294.52
28			1321.19
28			1346.15
28	21	1.33	1369.57
28	22	1.27	1391.57
28	23	1.22	1412.28
28			1431.82
28			1450.28
28			1467.74
28			1484.28
28	28	1.00	1500.00

Figure 5. Path lenght misbalancing influence over clock speed.

As shown in previous table, path length doesn't have any influence in final clock speed (for example, table shows long path of 28 stages in all the cases). Smallest combinational path is the real limitating factor.

Achronix ACE tool provides a synthesis directive named "set-extra-pipeline" that allows to automatically fill underfilled pipelines paths, to tune the number of added dummy stages in short length path, to increase maximum clock speed.

V. Aplication under Study

Due to the increasing of the AER community that develops AER devices to perform filters and transformation of AER coded visual information, some devices that generates AER events are needed in order to provide a stimulus to these filters.

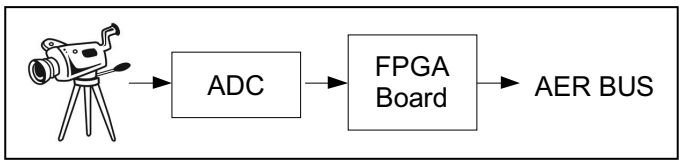


Figure 6. Synthetic retina boards

Figure 6 shows a block diagram of the proposed design. In figure 7, a USB-AER board (green board) is connected (input port) to an ADC board (copper) with video in connector, attached to a small B&W video camera.

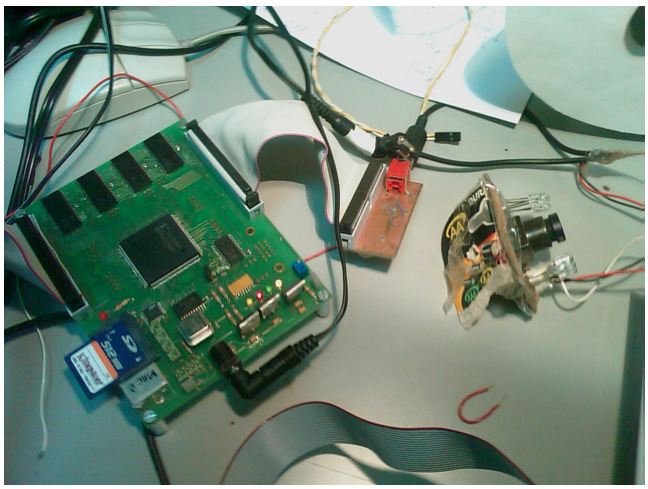


Figure 7. Synthetic retina boards

A. *Video Composite Capture.*

First step in frames to AER conversion rely on video capture sampling video composite signal provided by a camcorder or video camera. This camera should provide video composite signal. Like current AER filters works with monochrome images, and previously cited retinas are also monochromes, we are not going to process color signals.

For image capturing, a fast ADC manufactured by Analog Devices, ADC08100 is used. This ADC works with a clock signal between 20Mhz and 100Mhz providing up to 100MSPS (million of samples per second), delayed three clock cycles. It outputs digital data as an 8-bit parallel word that is connected to input pins of FPGA. Clock signal provided to ADC (25MHz) is obtained from internal FPGA clock (50MHz divided by 2). This sample rate is enough to sample both, color and BW signal. A sample rate of about 4Mhz should be enough to sample BW signals, but ADC 08100 doesn't work below 20Mhz minimum frequency.

To remove color subcarrier external LC filter can be used to filter it, or a simple FIR (Finite Impulse Response) or IIR (Infinite Impulse Response) digital filter can be implemented inside FPGA to digitally filter it. Oversampling the video signal at 25 Mhz can be useful to perform mentioned digital filtering. For reducing FPGA usage, implementing FIR or IIR filter requires multiplication computation. Multiplication operations, needed for this filter implementation, require a big FPGA usage. Due to limited amount of resources available in SPARTAN II 200 FPGA used in USB-AER board [10], external LC filter are preferred and extra samples are simply discarded.

B. Intensity retina and derivative retina.

Once the current frame is captured and stored in memory, the AER event generator uses this data stored in memory to scan it and generate events properly. In current AER state of the art, two different kind of synthetic AER retina has been developed: intensity and derivative one.

1) Synthetic Intensity AER retina

This synthetic AER retina sends AER events that describe light intensity information of its pixels, so event frequency for a given pixel is proportional to the amount of light that this pixel of the sensor receives.

Figure 9 shows the block diagram for intensity synthetic retina. "Scan counter" is divided in two halves, row counter (less significant bits) and column counter (most significant bits). Content of scan counter is used to access image RAM and get pixel value. Data read from RAM is compared with frame counter to determine if event address has to be sent. If comparison results positive, it enables that the value of scan counter used for pixel addressing, is now used as event address and it is loaded to a FIFO in order to be sent out through an AER output finite states machine.

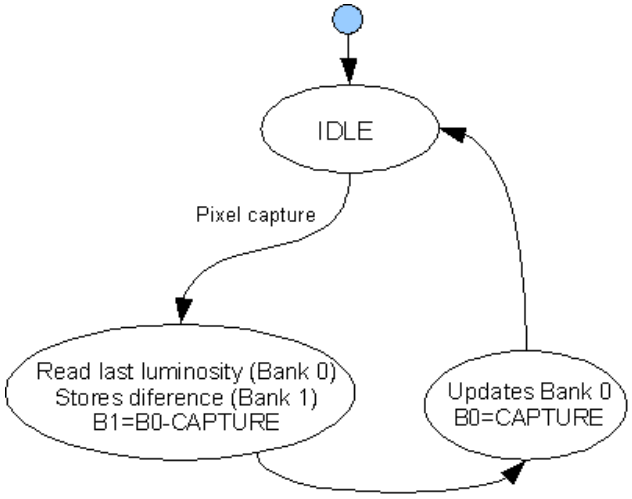


Figure 8. Synthetic retina hardware implementation state machine.

Frame counter is increased every time scan counter overflows, so they both can be implemented together, placing frame counter as the top most significant part in the combined counter. Frame counter bits are position reversed prior to comparison, as described later to enhance event time distribution.

Video composite capturer uses horizontal and vertical sync pulses to maintain capture line counter synchronized to current video line transmitted on video composite signal. Horizontal scan timer is synchronized with horizontal sync pulses to correctly temporize capture instants. It is implemented as another finite states machine. This states machine is shown in figure 8.

2) Synthetic Derivative AER retina

In this synthetic AER retinas, event frequency is not proportional the amount of light received, but the difference of light between current captured frame and last one, so event frequency is proportional to the change of luminosity. Memory requirements for this kind of retina is twice than for intensity one because it is necessary to store two different frame information, current one and previous one so we can subtract them. Alternatively, it is also possible to implement derivative retina storing last frame and the difference between it and current. Difference frame is used to generate AER events, and last frame intensity information is used to calculate difference frame when next frame is going to be captured.

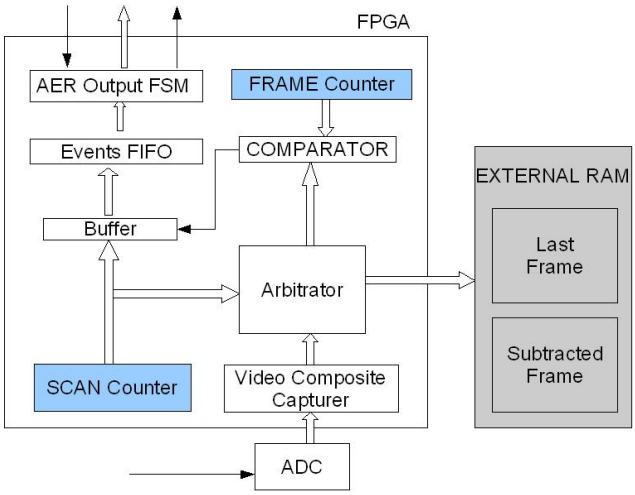


Figure 9. Derivative retina block diagram.

The result of subtracting two frame can have positive or negative sign, depending on pixel luminosity change: from darker to lighter, or opposite. In order to correctly represent sign of subtraction when events are generated, event frequency is proportional to absolute value of subtraction, but event address is different if sign is positive or negative.

Derivative retinas can be useful because the AER traffic generated is lower than brightness retina. Images with no variation or small variations between frames generate low event activity. Only those parts of the images that changes generates AER events. This can be used to easily track moving objects in a fixed background.

For external RAM, only one access bus is available (are implemented using fast static one port RAM chips), so an arbiter is needed to manage access to RAM from video composite capturer and AER event generator. The number of RAM access from event generator are several times bigger than access from video capturer, so usually RAM chips are assigned to it, except when a new pixel is captured, that momentary interrupts the generator.

VI. COMPARISON STUDY

A. Maximun clock frequency comparison

In this section we compare maximum clock speeds that can be reached for each device synthesis. At Xilinx Spartan II initial design the input clock (50MHz) is doubled by an internal PLL (100MHz) to be used in some parts of the design in order to improve performance. As can be seen in table II, this

100MHz clock cannot be used for the entire design because there are some parts of the circuit that requires a slower clock.

TABLE I. MAXIMUM CLOCK SPEED

FPGA Device	Synthesis tool	Maximum Frequency
Xilinx Spartan xc2s200-6	Xilinx ISE 10.1	68 Mhz
Xilinx Virtex 5vlx30ff324-3	Xilinx ISE 10.1	203Mhz
Achronix Speedster ACSPD60-FBGA1892	Synopsys Simplify PRO & Achronix ACE	545Mhz [a]

a. Depends of extra pipeline parameter

Achronix asynchronous design claims that can easily handle and combine different clock domains in the same design. Anyway, in this paper we are focused just in comparing FPGA performance for a non trivial design, which implements a commonly used finite state machine, where state transition requires information about input and current state information. This minimizes pipelining capabilities because of feedback loops but give us more realistic information about logic performance than only feed forward process. Feed forward clock speed can be increased just adding more and more pipeline stages in the design that can provide to us very high clocks speed, but it is not accompanied by real improvements in system performance (throughput of the system), as can be seen in table III.

TABLE II. EXTRA PIPELINING VALUES VS. CLOCK FREQUENCY

Variation of XP for clk		
Extra Pipelining [a]	Period (ps)	Frequency (Mhz)
0	26388.0	37.9
1	13194.0	75.79
2	6597.0	151.6
3	4398.0	227.4
4	3295.0	303.2
5	2638.0	379.0
6	2199.0	454.8
7	1884.9	530.5
8	1835.0	545.0

a. Set by "set-extra-pipeline" directive

Based on these syntheses, we have measured the number of events per second provided by the system, to get a practical measurement of performance. Due to internal fifos that divides performance influence between different parts of the global design, and influence of represented image in the number of events sent through the AER bus, we have supposed that all pixel has maximum activity so the number of event transmitted are maximum.

To indicate system latency, not all design is taken into consideration. Video composite capture is independent of event generation algorithm. A given pixel luminosity value is not determined by just one event transmission, and event transmission is homogeneously distributed in time, so it is impossible to measure latency between the moment a pixel value is read by the system, and the instant that it is sent out. Instead of it, latency is estimated between the moment a pixel value is read from RAM memory, and supposing it will be sent, the moment that the event comes out.

TABLE III. AER EVENT THROUGHTPUT

FPGA Device	Million events per second	Estimated latency
Xilinx Spartan xc2s200-6	17 Mev/s	140 ns
Xilinx Virtex 5vlx30ff324-3	50 Mev/s	49 ns
Achronix Speedster ACSPD60-FBGA1892	136Mev/s	34 ns

B. Synthesis time comparison

In this section we compare the time consumed by the synthesis tool (synthesis software) to complete synthesis, placement and optimization processes. In order to make comparable these time values we have used for the three cases the same computer (Intel Dual Core processor with 2GB RAM). If this process is repeated on another computer with different clock speed, or memory size, different values can be obtained, but absolute values won't be different. Therefore, these synthesis times can be used as a metric to compare how much faster is a software than the other for a fixed hardware configuration.

TABLE IV. SYNTHESIS TIME COMPARISON

FPGA Device	Synthesis tool	Synthesis Time[a]
Xilinx Spartan xc2s200-6	Xilinx ISE 10.1	0' 56"
Xilinx Virtex 5vlx30ff324-3	Xilinx ISE 10.1	2' 26"
Achronix Speedster ACSPD60-FBGA1892	Synopsys Simplify PRO & Achronix ACE	2' 10"

C. Resource usage comparison

In this section we compare resources usage for each FPGA device. Resources usage is not an easy parameter to be measured. In both, Achronix and Xilinx devices, logics are grouped in CLB or picoPIPEs, so it is not possible to directly extrapolate resource consumption, for example traducing the design requirements in a standard parameter as can be number of transistors used in each case. Even in Xilinx devices, CLB composition varies from one family of FPGA to another. Even knowing the number of transistor implied in each CLB, not all logic in a CLB can be completely used due to CLB organization, so some resources are unusable after doing placement.

TABLE V. RESOURCE USAGE COMPARISON

FPGA Device	Synthesis tool	Logic Slices Usage
Xilinx Spartan xc2s200-6	Xilinx ISE 10.1	7% (174/2352)
Xilinx Virtex 5vlx30ff324-3	Xilinx ISE 10.1	2% (105/4800)
Achronix Speedster ACSPD60-FBGA1892	Synopsys Simplify PRO & Achronix ACE	0.42% (196/47040)

Anyway, we consider that it can be an interesting parameter to compare qualitatively design complexity and FPGA capabilities to fit complex designs. It can be used as a metric for estimating proportionally the size of a given FPGA comparing to another, regardless the ambiguous parameter of FPGA size given in number of CLBs or picoPIPEs available, not knowing how big or how complex is one of these blocks. In table IV we show the resources consumption for the same design in these three FPGAs. It can be seen that the number of slices used for the Virtex 5 is lower that for Sparta II, and that Achronix requires more PicoPipes that slices in any Xilinx FPGA.

VII. CONCLUSIONS

In this paper we have described a design based of FPGA logic programmable device, used in spiking systems for real time image processing. In this case, AER device described is a synthetic AER retina emulator, used to simulate spiking retina behavior getting as video source a standard video composite source.

This design has been synthesized in synchronous and asyncrhonous FPGA devices to compare their capabilities. For that we have used two Xilinx FPGAs (Spartan II and Virtex 5 families) and an Achronix Speedster FPGA. We have used previous mentioned design as a benchmark to compare size, speed and capabilities of these FPGAs devices.

Achronix ACE synthesis tool error report is not as intuitive as Xilinx is, and also, required design practices are more restrictive that when designing for Xilinx. Anyway, this is due that Xilinx development environment is implanted long time ago, and currently they deployed version 13, while Achronix has only a few years of presence in the market, and is not widely used. Sometimes Achronix tools can crash due to incorrectly specified designs. Anyway, when design is correctly specified taking Achronix recommendations into account and synthesis success, final design clock speed are very promising. We have to take also in consideration, that synthesis effort required to the tool to automatically translate synchronous design to innovative asynchronous internal architecture of asynchronous FPGA makes more complex synthesis task.

If not using properly pipeline capabilities, Achronix performance is not very significant, being below cheapest Xilinx Spartan II results, but with proper tuning, obtained clock frequency multiplies several times clock frequency obtained with Xilinx top end families (2.5x in own case, up to 5x claimed by Achronix). Latency time is affected by pipeline stages, but global performance and throughput is improved.

ACKNOWLEDGMENT

This work has been supported by Spanish VULCANO project (TEC2009-10639-C04-02) funded by the Minister of Science and Innovation of the Government of SPAIN. We also want to thank E-Lab from Yale University and Professor Dr. Eugenio Culurciello for inviting us to his lab and give to us the opportunity to use the Achronix Speedster SPD60 evaluation board and Ken Nechamkin from Achronix Semiconductor for their technical support.

REFERENCES

[1] P. Lichtsteiner, C. Posh, T. Delbruck., "A 128×128 120dB 15 us Asynchronous Temporal Contrast Vision Sensor"IEEE Journal on Solid-State Circuits, vol. 43, No 2, pp. 566-576, Feb-2008.

[2] E. Culurciello, R. Etienne-Cumming, and K.A. Boahen, " A biomorphic digital image sensor" IEEEE journal of Solid-State Circuits, vol. 38, p 281-204, 2003

[3] V. Chan, S.C. Liu, A.van Schaik, "AER EAR: A Matched Silicon Cochlea Pair with Address-Event-Representation Interface". IEEE Transactions on Circuits and Systems-I. Vol. 54, No 1. pp. 48-59. Jan-2007.

[4] Teresa Serrano-Gotarredona, Andreas G. Andreou, Bernabé Linares-Barranco. "AER Image Filtering Architecture for Vision-Processing Systems". IEEE Transactions on Circuits and Systems. Fundamental Theory and Applications, Vol. 46, N0. 9, September 1999.

[5] Oster, M.; Douglas, R.; Shih-Chii Liu, "Quantifying Input and Output Spike Statistics of a Winner-Take-All Network in a Vision System" Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on 27-30 May 2007 Page(s):853 – 856

[6] P. Hafliger. "Adaptive WTA with an Analog VLSI Neuromorphic Learning Chip". IEEE Transactions on Neural Networks, vol. 18, No 2, pp. 551-572. March-2007.

[7] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, C. Serrano-Gotarredona, J.A. Perez-Carrasco, B. Linares-Barranco, A. Linares-Barranco, G. Jimenez, A. Civit. "On Real-Time AER 2-D Convolutions Hardware for Neuromorphic Spike-Based Cortical Processing. IEEE Transactions on Neural Networks, Vol. 19, No 7, pp. 1196-1219. July-2008.

[8] F. Gomez-Rodríguez, A. Linares-Barranco, L. Miró, S.C. Liu, A. van Schaik, R. Etienne-Cummings, M.A. Lewis, "AER Auditory Filtering and CPG for Robot Control". ISCAS 2007.

[9] R. J. Vogelstein, R. Etienne-Cummings, N.V. Thakor, A. H. Cohen. "Dinamic Control of Spinal Locomotion Circuits". ISCAS 2007.

[10] Gomez-Rodriguez, F.; Paz, R.; Linares-Barranco, A.; Rivas, M.; Miro, L.; Vicente, S.; Jimenez, G.; Civit, A.; "AER tools for communications and debugging" Circuits and Systems, 2006. Proceedings. 2006 IEEE International Symposium on 21-24 May 2006. ISCAS.2006.

[11] R. Paz-Vicente, A. Linares-Barranco, D. Cascado, S. Vicente, G. Jimenez, A. Civit. "PCI-AER interface for Neuro-inspired Spiking Systems". ISCAS 2006. Kos, Greece.

[12] Perez-Carrasco, J.A.; Serrano-Gotarredona, T.; Serrano-Gotarredona, C.; Acha, B.; Linares-Barranco, B.; "High-speed character recognition system based on a complex h ierarchical AER architecture" Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on 18-21 May 2008 Page(s):2150 – 2153. ISCAS.2008

[13] R. Berner, Tobi Delbrück, Antón Civit Balcells, Alejandro Linares-Barranco. "A 5 Meps $100 USB2.0 Address-Event Monitor-Sequencer Interface". ISCAS 2007.

[14] R. Serrano-Gotarredona , M. Oster, P.. Lichtsteiner, A. Linares-Barranco, R. Paz, F. Gomez-Rodriguez, H. Kolle Riis, T. Delbrück, S.C. Liu, S. Zahnd, A.M. Whatley, R. Douglas, P. Häfliger, G. Jimenez, A.

Civit, T. Serrano-Gotarredona, A. Acosta, B. Linares-Barrancoet al. “AER Building Blocks for Multi-Layer Multi-Chip Neuromorphic Vision Systems”. NIPS 2005.

[15] Kwabena A. Boahen. “Communicating Neuronal Ensembles between Neuromorphic Chips”. Neuromorphic Systems. Kluwer Academic Publishers, Boston 1998.

[16] A. Jiménez-Fernández, A. Linares-Barranco, R. Paz-Vicente,C.D. Luján-Martínez, G. Jiménez, A. Civit. “AER and dynamic systems co-simulation over Simulink with Xilinx System Generator”. ICECS 2008

[17] M. Sivilotti, Wiring Considerations in analog VLSI Systems with Application to Field-Programmable Networks, Ph.D. Thesis, California Institute of Technology, Pasadena CA, 1991.

[18] A. Linares-Barranco, R. Paz-Vicente, G. Jimenez, J.L. Pedreño-Molina, J. Molina-Vilaplana, J.L. Coronado et al.. “AER Neuro-Inspired interface to Anthropomorphic Robotic Hand”. IEEE World Conference on Computational Intelligence. IJCNN. Vancouver, July-2006.

[19] A. Jiménez-Fernández, R. Paz-Vicente, M. Rivas, A. Linares-Barranco, G. Jiménez, A. Civil. “AER-based robotic closed-loop control system”. ISCAS 2008.

[20] F. Gomez-Rodriguez, R. Paz, L. Miro-Amarante, Alejandro Linares-Barranco, Gabriel Jiménez, “Two Hardware Implementation of the Exhaustive Synthetic Aer Generation Method”. LNCS 2005.

[21] jAER software. http://jaer.wiki.sourceforge.net/