SHORES: Software and Hardware Open Repository for Embedded Systems

Laurentiu Acasandrei, and Angel Barriga, *Member, IAENG*

***Abstract*— This communication describes an open source repository for embedded software and hardware designs. Its main goal is to make available to everyone, in an open-source style, the designs and results from academia/research community. SHORES hosts the source code of various software and hardware design projects, that combined with the newest algorithms proposed by academia, give birth to embedded solutions to the most challenging obstacles in the fields of vision, bio-cryptography, signal processing, etc. SHORES resources are distribute under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the license, or any later version.**

***Index Terms*—Embedded system, intellectual property module, open source hardware, open source software.**

I. Introduction

The open source philosophy is well known in software, and has been exploited extensively. However the concept of open source hardware is more recent. The Open Source Hardware Association [1] defined the open source hardware (OSHW) as "*a term for tangible artifacts (machines, devices, or other physical things) whose design has been released to the public in such a way that anyone can make, modify, distribute, and use those things. This definition is intended to help provide guidelines for the development and evaluation of licenses for Open Source Hardware*".

There are different types of OSHW products. One type of OSHW products are electronic artefacts that are distributed so that the user can access to all the information for implementation such as the schematic, system specifications, component distributers, etc. An example of this type of product is Open Electronics [2]. For other type of OSHW products the development tools are provided. A well-known example is Arduino [3].

Another type of OSHW product corresponds to IP (Intellectual Property) modules described in a hardware description language. These modules can be used and modified by the user to implement a specific application. The rise of this type of product began with the development of FPGA devices.

Manuscript received May 10th, 2017; revised June 8th, 2017 This work was supported in part by Ministerio de Economía y Competitividad under the Projects TEC2014-57971-R, co-financed by FEDER.

Laurentiu Acasandrei is with the Instituto de Microelectrónica de Sevilla, IMSE-CNM, CSIC/Universidad de Sevilla, Spain (e-mail: laurentiu@imse-cnm.csic.es). Angel Barriga is with the Instituto de Microelectrónica de Sevilla, IMSE-CNM, CSIC/Universidad de Sevilla, Spain (e-mail: barriga@imse-cnm.csic.es).

A. *OSHW sites*

There are few open source hardware products compared to the case of software. Resources exist primarily oriented to specific applications [4-7].

There are few repositories that offer multiple open source hardware projects. Table I shows a list (not intended to be complete) of some of the main repositories.

OpenCores [8] is the most known site of open source hardware IP cores. It host a large set of source code for different digital hardware projects (IP-cores, System on Chip (SoC), boards, etc) and support the users with different tools, platforms, forums and other useful information. One useful resource of this site is the forums in which users can interact for help, exchange ideas, knowledge and experience. This active site is the starting point for searching open source hardware for any designer.

Freecores [9] contain cores most of which are in OpenCores. The main useful of this site is to provide the benefit of using *git* (a distributed revision control systems).

Open Hardware Repository [10] at CERN is a site containing open hardware resources at experimental physics applications. This site offers two kinds of facilities: open hardware projects and support (containing information for users).

The BeagleBoard.org Foundation [11] is a non-profit corporation existing to provide education and promotion of the design and use of open source software and hardware in embedded computing. BeagleBoard.org provides a communication forum for the owners and developers of open-source software and hardware.

Table I. OSHW sites

Name	Ref	# Projects
OpenCores	[8]	1241
Freecores	[9]	768
OHWR	[10]	100
BeagleBoard.org Foundation	[11]	484
Zoybar	[12]	25
SHORES	[13]	10
Free Model Foundry	[14]	> 11,000 components
TimVideos	[16]	4
Milkymist	[17]	4
MinSoC	[18]	1

Other examples correspond to IP modules of processors (and associated peripherals) that are distributed under the concept of open source hardware. Here we highlight the case of Leon [18], OpenRisc [8], OpenSparc [19], S1 Core [20], F-cpu [21], etc.

For the case of open source software there are numerous sites for embedded software applications. Some examples are: OpenEmbedded [22], SourceForge [23], Open Embedded Software Foundation [24], Embedded [25], etc.

B. *Licensing issues*

An important issue to consider for open source is the license aspect. In this case, OSHW inherits the open source software licenses scheme. According to [1] "*in general, there are two broad classes of open-source licenses: copyleft and permissive. Copyleft licenses (also referred to as "share-alike" or "viral") are those which require derivative works to be released under the same license as the original; common copyleft licenses include the GNU General Public License (GPL) and the Creative Commons Attribution-ShareAlike license. Other copyleft licenses have been specifically designed for hardware; they include the CERN Open Hardware License (OHL) and the TAPR Open Hardware License (OHL). Permissive licenses are those which allow for proprietary (closed) derivatives; they include the FreeBSD license, the MIT license, and the Creative Commons Attribution license. Licenses that prevent commercial use are not compatible with open-source*". The GNU Lesser General Public License (LGPL) is a non-viral license, instead GPL which is viral, widely used.

Open source concept means that the source code is publicly available under a license that gives users the right to change, and distribute the software. The term was coined in 1998 by Open Source Initiative (OSI). The "free" of open source refers exclusively to the source code, and it is possible to make business giving support, services, documentation, and even binary versions.

All the free software can be described as open source, however almost all open source software is free software. The difference is subtle but in the case of hardware the term "open source" is more appropriate since the sources, schematic, etc., are offered, and the user is who makes the implementation. Therefore, OSHW is closer to the concept Do-It-Yourself (DIY) [26].

Another issue that may raise doubts in the field of open source refers to the authors copyright. As described in [27] there are many possible scenarios in the case of open source projects. In any case OSHW, like as in software, are protected by copyright laws. For open source the author is granting the user a license of the copyrighted hardware. The copyright and author rights of the work are given to the author whatever are the terms of the license.

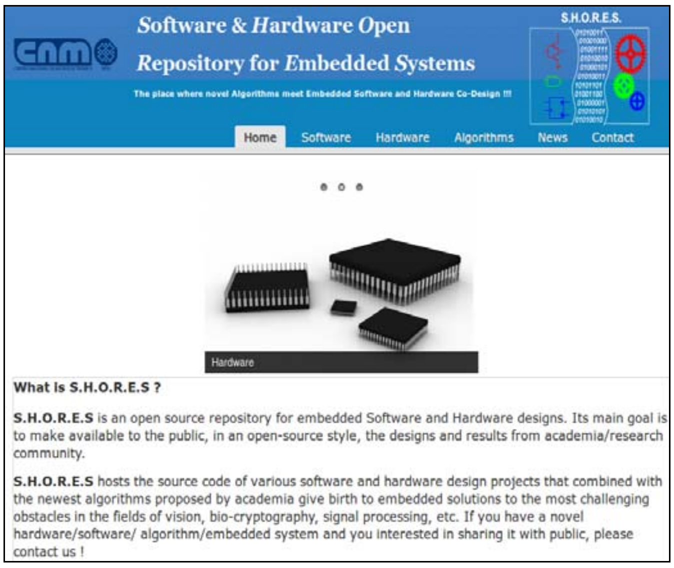
C. *SHORES site*

Software and Hardware Open Repository for Embedded Systems (SHORES) is an open source repository for embedded Software and Hardware designs. Its main goal is to make available to everyone, in an open-source style, the designs and results from academia/research community. SHORES hosts the source code of various software and hardware design projects that combined with the newest algorithms proposed by academia give birth to embedded solutions to the most challenging obstacles in the fields of vision, bio-cryptography, signal processing, etc.

II. SHORES Repository Description

Figure 1 shows the home page of the repository. The repository is organized into two main areas: Software and Hardware. There is also planned an area of algorithms that currently has no content. In each area the different projects are organized into categories.

An important characteristic of the projects included in the repository is that they have been tested in specific applications. This means that these are projects that have been proven and, therefore, are operational on the platform indicated in the project description.


Software & Hardware Open
Repository for Embedded Systems
The place where novel Algorithms meet Embedded Software and Hardware Co-Design !!!
S.H.O.R.E.S.
Home
Software
Hardware
Algorithms
News
Contact
Hardware

What is S.H.O.R.E.S ?

S.H.O.R.E.S is an open source repository for embedded Software and Hardware designs. Its main goal is to make available to the public, in an open-source style, the designs and results from academia/research community.

S.H.O.R.E.S hosts the source code of various software and hardware design projects that combined with the newest algorithms proposed by academia give birth to embedded solutions to the most challenging obstacles in the fields of vision, bio-cryptography, signal processing, etc. If you have a novel hardware/software/ algorithm/embedded system and you interested in sharing it with public, please contact us !

Fig. 1. SHORES home page.

The Software area of SHORES site contains embedded software projects. They are embedded applications running on a specific processor indicated in the project description. In general, software applications have been written to make them independent of the execution platform. Therefore the direct portability to other platforms should be feasible. Source files and documentation for the user to modify and adapt them to their needs are supplied.

Currently the software projects are focused within the category of Embedded Vision (EV). There are 3 projects related to detection and recognition. In particular these focus on face detection and recognition, but can be applied to the detection and recognition of objects in general.

Hardware area contains designs of hardware IPs organized into four categories: Communication Interfaces (CI), Embedded Vision (EV), Arithmetic Circuits (AC) and Video Coder/Encoder (VC). The circuits are described in VHDL and are portable to various FPGA and ASIC technologies.

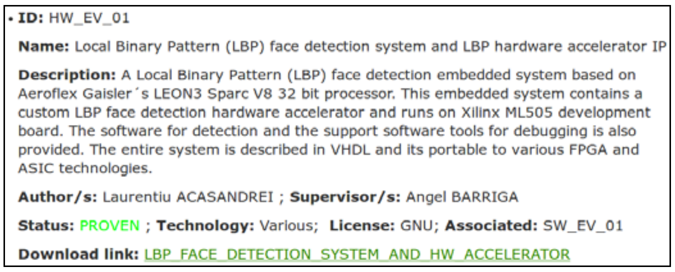
The information displayed for each project aims to identify functionality, design status, dependence with other projects and associated references. Figure 2 shows an example of a project. Each project has an ID that identifies it. ID consists of three fields. The first indicates the type of project: SW (Software) or HW (Hardware). The second field indicates the category in which the project is classified. For example in the case of Figure 2 the category is EV (Embedded Vision). The third field is a number that identifies the project.

The name and project description specifies its functionality, main characteristics, and the platform for which the application was designed and proven.

The status indicates that the project has been verified on the targeted hardware/software platform. All projects of SHORES repository must have the "Proven" status value. This ensures that the project has been tested and is operational. The operation of the project is guaranteed by the platform listed in the description.

Other information provided in the project is the "Associated" section. This section lists the prerequisites needed to implement the project. In the example of Figure 2 the execution platform based on Leon3 processor is required to run the software application.

There is also useful information that can be find in the list of publications related to the project. The publications describes algorithmic and implementation challenges/solutions. They complement the documentation of the project.


• **ID:** HW_EV_01

Name: Local Binary Pattern (LBP) face detection system and LBP hardware accelerator IP

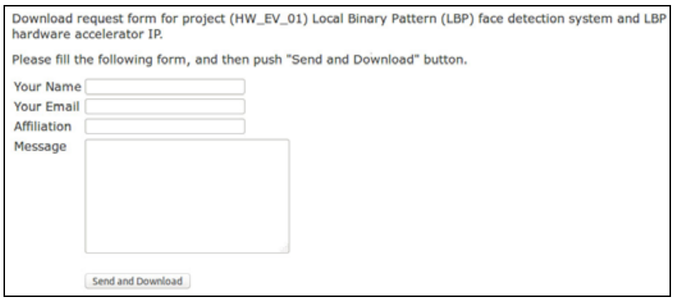
Description: A Local Binary Pattern (LBP) face detection embedded system based on Aeroflex Gaisler´s LEON3 Sparc V8 32 bit processor. This embedded system contains a custom LBP face detection hardware accelerator and runs on Xilinx ML505 development board. The software for detection and the support software tools for debugging is also provided. The entire system is described in VHDL and its portable to various FPGA and ASIC technologies.

Author/s: Laurentiu ACASANDREI ; **Supervisor/s:** Angel BARRIGA

Status: PROVEN ; **Technology:** Various; **License:** GNU; **Associated:** SW_EV_01

Download link: LBP_FACE_DETECTION_SYSTEM_AND_HW_ACCELERATOR

Fig. 2. Example of a project in SHORES.


Download request form for project (HW_EV_01) Local Binary Pattern (LBP) face detection system and LBP hardware accelerator IP.

Please fill the following form, and then push "Send and Download" button.

Your Name	
Your Email	
Affiliation	
Message	

Fig. 3. User form for download.

Each project has a download link. There is no need for user registration/authentication, but when activating the download link, a small amount of information is requested before the download is started. Figure 3 shows the form of user information. The information on this form is required for statistical purposes. It helps to know statistic information such as the geographic areas, universities or research centers that intend to use the applications/designs offered by the repository. It can be useful also to warn of new updates.

By activating the "Send and Download" button a download of the project is started. The downloaded file is a compressed file (zip file) containing the sources and documentation.

III. Software Projects

This section contains a brief description of the software projects, the authors, the status, an archive of the source code and archives, when is the case, with the auxiliary software or test benches used for debugging and verification.

Software section currently has a single category of applications corresponding to Embedded Vision. Within this category there are available three applications described below. Figure 4 shows the block diagram of a face identification system.

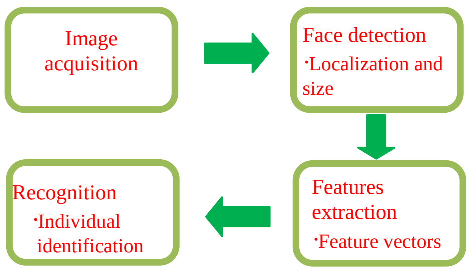


Fig. 4. Face identification system.

The starting point for the design methodology of the embedded system is the OpenCV's baseline face detection/recognition applications. OpenCV (OPEN source Computer Vision), started by Intel in 1999, is a library of programming functions for real time computer vision [28]. OpenCV is released under a BSD license and hence it is free for both academic and commercial use. It is written in C/C++ and was designed for computational efficiency and with a strong focus on real-time applications.

The host target for the proposed face detection system is an embedded environment based on LEON3 AMBA Bus processor. The LEON3 is a synthesizable VHDL soft core of a 32-bit processor compliant with the SPARC V8 architecture [18] [29]. The processor is highly configurable, and particularly suitable for System-on-Chip (SoC) designs. The full source code is available under the GNU GPL license. The processor controls and executes the majority of software application tasks while a specific IP module accelerate only those tasks that require a high number of clock cycles.

Each project includes two applications: OpenCV ported software version and algorithmic accelerated version application. For the OpenCV ported software version for embedded system it has been considered that the majority of embedded environments are capable of running C applications with or without operating system support. This means that the resulting application code has to be compatible for C compilers, and in the same time platform independent.

Another consideration made is the fact that most of the SoC have no floating point support. For it, the resulting application uses integer operations instead of floating point operations in order to preserve the generality of the application for the embedded system world. An important moment in this step was finding an acceptable scaling coefficient of the floating point variables and data to integer variables and data. After trying different values and comparing the resulted integer application with the floating point application, we found that by scaling with 20 bits (precision of 20 bits for the floating point decimals) the integer and floating point applications obtain identical results. Also the floating point squared root function, necessary to calculate variance of the evaluating window, was replaced with a fast integer squared root version. Finally, it was obtained a face detection stand-alone application compatible with C using only integer type operations and data.

For the algorithmic accelerated version application, different detection modes were analyzed, in order to find the run time bottlenecks and optimize the detection. There have been implemented some improvements to optimize the algorithms.

A. *Embedded Local Binary Pattern face detection software application*

This project includes two applications: OpenCV ported software version and algorithmic accelerated version of Local Binary Pattern (LBP) face detection application [30]. The two applications run on Xilinx ML505 development board containing LEON3 Sparc V8 32 bits synthesized. To run the applications one needs to download the hardware platform from project HW_EV_01, and follow the project instructions.

B. *Viola Jones face detection software application*

This project includes two applications: OpenCV ported software version and algorithmic accelerated version of Viola Jones face detection application [31-33]. The two applications run on Xilinx ML505 development board containing LEON3 Sparc V8 32 bits synthesized. To run the applications one needs to download the hardware platform from project HW_EV_02, and follow the project instructions.

C. *Face recognition using Fisherfaces (LDA) and Eigenfaces (PCA)*

This project includes two applications: OpenCV ported software version of the face recognition using Fisherfaces (LDA) and Eigenfaces (PCA) [34-35]. This project is composed by a group of four software applications that are involved in training of new classification vectors, verification of the trained results and face recognition. The first application is *cv_open_core*, and contains the core functions and data used in the rest of the applications. The *make_training* uses an image face database to train new classification vectors (either Fisherfaces or Eigenfaces) and must be run in operative system that has OpenCV installed. The *c_opencv_test* application measures the recognition performances of the new trained classification vectors and must run in operative system that has OpenCV installed. The *face_classification_leon3* application uses the trained vectors to recognize users from images. The face classification application targets the LEON3 embedded system running on Xilinx ML505.

IV. HARDWARE PROJECTS

This section contains a brief description of the hardware design, the authors, the status, an archive of the design and test bench used for debugging and verification.

The section of hardware projects currently has four categories of applications described below.

A. *Communication Interfaces (CI)*

There is a project containing various AMBA bus interfaces. These interfaces can be used on systems with different restrictions (low power, low speed to high speed applications transmission). The project consists in a VHDL library of most common AMBA master and slaves, and a VHDL test framework with basic examples. The interfaces are: APB slave, AHB master/slave, AXI master/slave, AXI-Stream master/slave. The project also includes an APB slave and AHB master/slave test framework in order to help the verification of the use of the interfaces. As an example, Figure 5 shows the test bench setup for APB slaves. An application example of AXI-Stream bus is described in [36].

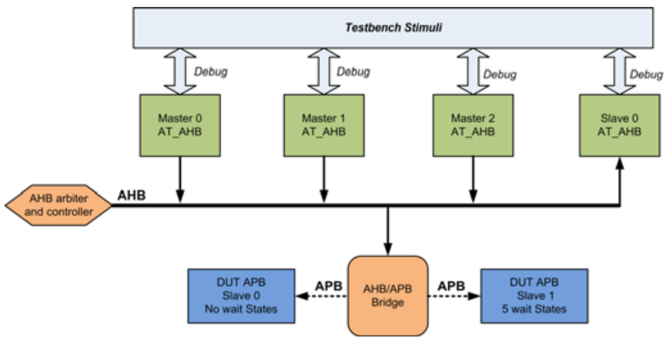


Fig. 5. AMBA testbench setup for APB slave

The AMBA architecture, as being an open standard, has the advantage that there are available many bridges to other communication architectures (Core Connect, Wishbone, Avalon, etc). This means that the interfaces developed in this project can be integrated with minimal effort in embedded systems based on different bus architectures.

B. *Embedded Vision (EV)*

In this category there are two projects related to face detection IP modules: A Local Binary Pattern (LBP) face detection embedded system and a Viola-Jones face detection embedded system [37-39].

Both projects are based on Aeroflex Gaisler´s LEON3 Sparc V8 32 bit processor. The embedded system projects contain a custom face detection hardware accelerator (based in LBP algorithm or in Viola-Jones algorithm, respectively) and run on Xilinx ML505 development board. The software for detection and the support software tools for debugging are also provided. The entire system is described in VHDL, and it is portable to various FPGA and ASIC technologies. Figure 6 shows the schema of the application system implementation.

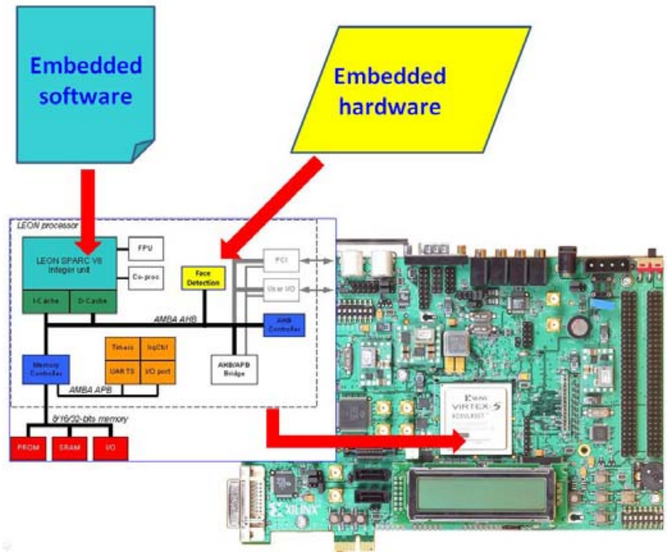


Fig. 6. Embedded face detection system implementation.

Based on the algorithmic accelerated software version application those parts that consume a many resources and have large run times were implemented as an IP circuit. The idea is to offload to the hardware the functions with a high degree of processing, and to parallelize the execution of the detection algorithm. With this, the face detection process can be drastically accelerated.

The proposed embedded software application is responsible for initializing the face detection IP memory (i.e. in the case of Viola-Jones algorithm it stores the Haar-like features in the IP shared memory). Also it initializes the face detection IP configuration registers. When the software application starts the face detection process, the IP module takes the control in order to search faces in variable size regions. Upon completion the detection process, the IP signalize if there are faces by updating the status register and generating an interrupt. The result of the face detection system consists of the coordinates of the face in the image, as well as its size. This information is read by the software application in order to be used for face recognition. Once finished the recognition, the system informs if the individual has been identified or not.

C. *Arithmetic Circuits (AC)*

The three projects under this category are arithmetic circuit generators. It was found that the number of free, publicly available arithmetic module generators offered from academia or industry to implement a fast multiplier is quite low. In academia the arithmetic module generator (ARITH) with algorithm optimization capability [40] can generate VHDL and Verilog descriptions for multipliers and adders. The ARITH generator [41] is capable of generating arithmetical blocks (multipliers and adders) having any width between 4 and 64. Another initiative is FloPoCo (FLOating-POint COres) which is a generator of arithmetic cores for FPGAs [42-43]. It generates VHDL synthesizable code of the arithmetic modules

The first project is a multiplier generator called SLAM (Scalable Low Area integer Multiplier generator). The scalable low area multiplier is a novel architecture targeting low power/area applications. It can generate integer signed/unsigned multipliers having operands with a data width between 8 and 64 bits.

```
entity SLAM_signed_multiplier_NxN is
   Generic (N: integer:= 8);
   Port ( rstn : in  STD_LOGIC;
          clk : in  STD_LOGIC;
          OPA : in  STD_LOGIC_VECTOR (N-1 downto 0);
          OPB : in  STD_LOGIC_VECTOR (N-1 downto 0);
           PROD : out  STD_LOGIC_VECTOR (2*N-1 downto 0));
end SLAM_signed_multiplier_NxN;
```

Fig. 7. Entity of NxN bit signed multiplier

The second project is a non-restoring integer square root pipelined generator. The generator targets high speed high throughput applications. It can generate integer square root circuits having the operand with a data width between 8 and 256 bits.

Finally the third project is a radix 4 sequential integer square root generator. The generator targets targeting low power/area applications. It can generate integer square root circuits having the operand with a data width between 16 and 256 bits.

D. *Video Coder/Encoder (VC)*

In this category there is one project named BT656 (ITU656) Video Stream Decoder. This AMBA bus compatible IP is used to decode video streams compressed with BT656 standard. This IP was implemented in embedded system containing LEON3 Sparc V8 processor connected to a Videology 21C405W camera [44]. The input video is decoded in real time and each frame is saved to a user defined memory area. The download link contains the VHDL sources code of the BT656_IP, the embedded system containing the IP and the software drivers, and software examples on how to use the IP.

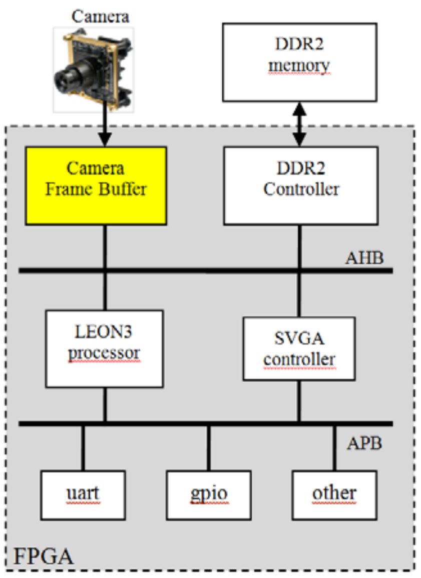


Fig. 8. Video Stream Decoder connected to camera and Leon3 system

V. Conclusions

An open source repository for embedded software and hardware applications is described. The hardware IP modules are described in a hardware description language (VHDL). The software applications are written in C language in a technology independent style. These modules can be used and modified by the user to implement their own specific application. The repository aim is to give support to the embedded system community in the design process, in order to increase productivity, and helping in the developing of complex systems.

References

[1] OSHWA: http://www.oshwa.org/
[2] Open Electronics: http://www.open-electronics.org/
[3] Arduino: http://www.arduino.cc/
[4] Hacom: http://www.hacom.net/
[5] Adafruit: http://www.adafruit.com/
[6] Kosagi: http://www.kosagi.com/
[7] Open Compute Project: http://www.opencompute.org/
[8] OpenCores: http://opencores.org/
[9] Freecores: http://freecores.github.io/
[10] OHWR: http://www.ohwr.org/
[11] BeagleBoard.org Foundation: http://beagleboard.org/
[12] Zoybar: http://www.zoybar.net/
[13] SHORES: http://www.imse-cnm.csic.es/shores/
[14] Free Model Foundry (FMF): http://www.freemodelfoundry.com/
[15] TimVideos: http://code.timvideos.us/
[16] Milkymist: http://m-labs.hk/
[17] MinSoC: http://www.minsoc.com/
[18] Leon: http://www.gaisler.com/
[19] OpenSparc:http://www.oracle.com/technetwork/systems/opensparc/index.html
[20] S1 Core: http://www.srisc.com/
[21] F-cpu: http://f-cpu.seul.org/
[22] OpenEmbedded: http://www.openembedded.org/
[23] SourceForge: http://sourceforge.net/
[24] Open Embedded Software Foundation: http://www.oesf.biz/
[25] Embedded: http://www.embedded.com/
[26] M. Wolf and S. McQuitty: "Understanding the do-it-yourself consumer: DIY motivations and outcomes". Springer AMS Rev., 1:154–170, (2011)
[27] O. Johnny, M. Miller and M. Webbink: "Copyright in Open Source Software – Understanding the Boundaries". International Free and Open Source Software Law Review, vol. 2, issue 1, pp 13 – 38, (2010)
[28] OpenCV: http://sourceforge.net/projects/opencvlibrary/
[29] J. Gaisler, S. Habinc, E. Catovic: GRLIB IP Library User's Manual, Gaisler Research, 2009.
[30] Acasandrei, L., Barriga, A.: Design Methodology for Face Detection Acceleration. 39th Annual Conference on IEEE Industrial Electronics Society (IECON), November 2013.
[31] Acasandrei, L., Barriga, A.: Embedded face detection implementation. IEEE International Conference of the Biometrics Special Interest Group (BIOSIG), September 2013.
[32] Berni, J. F., Acasandrei, L., Galan, R.C., Barriga, A., Vazquez, A.R.: Power-efficient focal-plane image representation for extraction of enriched Viola-Jones features. IEEE International Symposium on Circuits and Systems (ISCAS), pp. 3122–3125, 2012
[33] Acasandrei, L., Barriga, A.: Accelerating Viola-Jones face detection for embedded and SoC environments. 5th ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC), 2011.
[34] Acasandrei, L., Barriga, A.: Face Identification Implementation in a Standalone Embedded System. International Symposium on Industrial Electronics (ISIE), June 2014.
[35] Acasandrei, L., Rodríguez, M.Q., Ribes, A.R., Barriga, A.: Sistema empotrado reconfigurable para aplicaciones de identificación de caras. Jornada de Computacion Reconfigurable y Aplicaciones (JCRA), Madrid 2013.
[36] Fularz, M., Kraft, M., Kasinski, A., Acasandrei, L.: A hybrid system on chip solution for the detection and labeling of moving objects in video streams. Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), pp. 94–99, 2013
[37] Acasandrei, L., Barriga, A.: AMBA bus hardware accelerator IP for Viola–Jones face detection. IET Computers & Digital Techniques, vol. 7, no. 5, pp. 200–209, September, 2013
[38] Acasandrei, L., Barriga, A.: Implementación sobre FPGA de un sistema de detección de caras basado en LEON3. Proceedings of the XVIII International IBERCHIP Workshop, pp. 6-9, 2012
[39] Acasandrei, L., Barriga, A.: FPGA implementation of an embedded face detection system based on LEON3. International Conference on Image Processing, Computer Vision, and Pattern Recognition, Las Vegas, 2012
[40] Watanabe, Y.; Homma, N.; Aoki, T.; Higuchi, T.; 'Arithmetic module generator with algorithm optimization capability', IEEE International Symposium on Circuits and Systems, pp.1796-1799, 2008
[41] Arithmetic Description Language: ARITH. http://www.aoki.ecei.tohoku.ac. jp/arith/
[42] Florent de Dinechin and Bogdan Pasca. Designing custom arithmetic data paths with FloPoCo. IEEE Design & Test of Computers, August, 2011
[43] FloPoCo: http://flopoco.gforge.inria.fr/
[44] Videology Imaging Solutions, Inc.: http://www.videology.nl/