Baltic J. Modern Computing, Vol. 4 (2016), No. 1, 79-88

Energy Efficient Platform for Sobel Filter in Energy and Size Constrained Systems

Rokas JUREVIČIUS, Virginijus MARCINKEVIČIUS

Vilnius University, Institute of Mathematics and Informatics, Akademijos str. 4, LT-08663, Vilnius, Lithuania

{rokas.jurevicius,virginijus.marcinkevicius}@mii.vu.lt

Abstract. The design space of a computer vision engineer is very large when it comes to the selection of hardware for high performance and energy efficient computing. By comparing a few potential (*Parallella* with Epiphany co-processor, *Radxa Rock2* with Mali T764 GPU and *Airvision Core X1* with Nvidia Tegra X1) platforms we have narrowed down this design space. This paper analyses three recent heterogeneous platforms for a typical image processing application – convolution based *Sobel* filter. By measuring platforms energy consumption while computing processing intensive task of image filtering we are trying to identify best fit embedded heterogeneous computing platform for energy and sized constrained environment. Platforms were selected by the ability to incorporate parallel computing on a co-processor or a GPU, but also should use less than 10 Watts of electrical power and should be no larger than a credit card so it would be suitable for such constrained environments as small UAVs. The results shown that GPU platforms are more efficient compared with single core CPU application and co-processor technology, though Nvidia Tegra X1 processor was best performing (142x faster than a single core application and 29x faster than nearest GPU opponent) and the most energy efficient (used 84x less energy than a CPU and 12x than GPU opponent).

Keywords: image processing, energy efficient, parallel computing, embedded platform, benchmarking, Sobel filter.

1. Introduction

Sophisticated computer vision applications often used in robotics requires a lot of computing power, i.e. recent research demonstrates that vision-aided navigation is feasible during flight-time to operate in GPS-denied environments, but the challenge is in dealing with the power and weight restrictions on-board a UAV while providing necessary robustness (Chowdhary, 2013). Most modern computers have enough computing power to run complex computer vision algorithms, though such machines often consumes tenths or hundreds watts of electrical power and are large in size compared to small robotic systems or UAV's. The computing power requirements can be ignored until the system has to employ image processing and computer vision algorithms for obstacle avoidance, object recognition or vision-aided guidance using digital cameras (Uragun, 2011).

Researchers developing a framework for computer vision algorithm benchmark mentioned that in an energy and performance constrained context, such as a battery-

powered robot, it is important to achieve sufficient accuracy while maximising battery life (Nardi, 2014). By using less power on computations may allow the usage of more sensor which could increase aerial vehicle security and flight distances. This paper analyzes the implementation of a popular *Sobel* filter algorithm used in image processing with energy efficiency in mind.

Fig. 1. The Parallella platform.

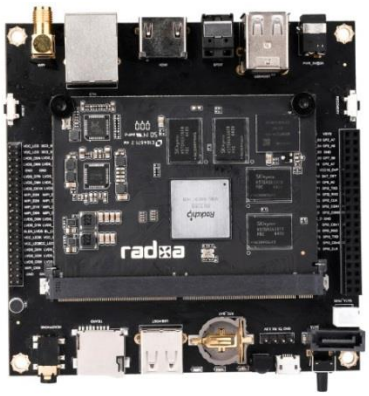

Fig. 2. Radxa Rock2 Square platform.

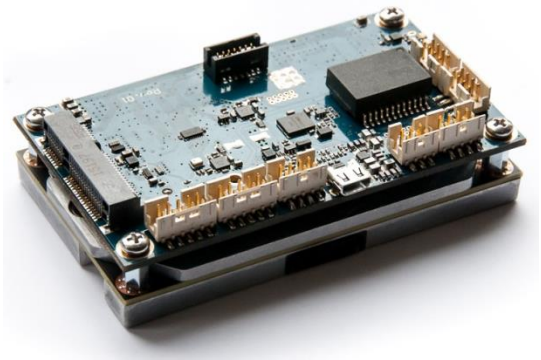

Fig. 3. Airvision Core X1 platform.

1.1. Related work

As described in (Fowers, 2012), there are three main high-performance platforms for image processing (specifically sliding-window algorithms): multi-core systems, GPU and FPGA. The research conducted in (Fowers, 2012) states, that multi-core CPU systems may be very energy inefficient compared with FPGA and GPU implementations. The new *Parallella* (see figure 1) platform may be of interest since it implements low power RISC architecture. The platform contains Xilinx Zynq-7010 processor with integrated FPGA programmable logic and a 16-core Epiphany co-processor. The platform uses FPGA logic only for data line interconnections between Xilinx CPU and Epiphany co-processor. The Epiphany co-processor was designed to execute parallel applications with high energy efficiency. The 64-core Epiphany processor variant has shown to have a great potential in energy efficient computing by achieving 2400 cycles per second per Watt (c/s/W) compared to 79 c/s/W for the Intel

i7-4770K CPU (Olofsson, 2014). So it should be possible to achieve the same computational power using 30 times less power. These numbers are very rough since a single cycle in Intel's instruction set may do such amounts of works which could require more then several cycles in Epiphany's RISC architecture. Still a research may be conducted to compare this platform with other efficient platforms - FPGA or an embedded GPU. I. Grasso has conducted a research of embedded Mali GPU performance and energy for high-performance computing (HPC), the potential is that Mali 604 GPU has a 8.4x computing speedup compared with Cortex A-15 CPU core while using 32% of the energy (Grasso, 2014). Previous research conducted by Josip Knezovic′ (2014) implementing a blow-fish password hacking algorithm has shown to be very efficient from energy perspective. The efficiency of the algorithm implementation was measured in password cracks per second per Watt of power. The Epiphany 16 core co-processor was able to crack as many passwords as Intel's T7200 processor, but epiphany processor requires 17 times less energy to do the same calculations (Knezovic, 2014). Comparison of efficiency of multi-core CPU, GPU and FPGA platforms in a real-world image processing application was the motivation for the experiments described in this paper. Three hardware platforms were chosen for a comparison. All of the platforms were chosen to be small, maximum power usage under 10 watts and small enough to fit most UAVs. One platform used in this research will be *Parallella* (WEB, g), which uses a 16 core RISC co-processor designed for parallel computing. Another platform will be *Radxa Rock2* SoM[1] (see figure 2), which employs Rockchip RK3288 quad-core ARM CPU and Mali T764 GPU. The Mali GPU is one of the few widely available embedded GPU cores with OpenCL 1.1(WEB, b) support for general purpose parallel computing. The third is *Airvision Core X1* (WEB, a) which provides real-time computer vision and navigation for Unmanned Aerial Vehicles (UAV) (see figure 3). This hardware platform uses Nvidia Tegra X1(WEB, f) chip with ARM quad-core CPU and Maxwell architecture GPU with 256 cores and is programmable using CUDA toolkit (WEB, e).

1.2. Parallel implementation of the Sobel filter algorithm

Parallel programming will be used to employ all of the 16 cores of the P*arallella* platform. The software can be developed eSDK (Epiphany SDK, developed by Adapteva (WEB, d)) or OpenCL implementation for the Epiphany processor by the Brown Deer Technology (OpenCL, part of COPRTHR framework (WEB, c)). The software for Mali GPU was developed using OpenCL framework. Meanwhile Airvision Core X1 development board will be programmed using Nvidia CUDA toolkit (WEB, e). A data parallelization technique was chosen for the implementation which is described in OpenCL introduction (Tompson, 2012) - the input data is divided into even sub-arrays, sub-array count is selected by the computational core count in the accelerating hardware.

[1] Radxa Rock2 SoM, manufacturer site, available online: http://radxa.com/Rock2/som

a) Input image b) Vertically filtered image c) Horizontally filtered image

Fig. 4. Input and output images of the *Sobel* filter.

An evaluation of the two frameworks and platforms was carried out by implementing *Sobel* filter (Pingle, 1969) algorithm. Two convolutions of the input image was calculated using frame

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A$$

for vertical image convolution and

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A$$

for horizontal image convolution (Pingle, 1969). In the convolution frame equations '$*$' symbol notes convolution operation with 2D pixel array A. Input image example is available in figure 4a, also output horizontal and vertical images are available in figure 4b and 4c respectfully. The convolution will be calculated for 100 iterations using the same image and the execution time (in seconds) for each frame will be measured. For a more real-life evaluation and in-depth execution analysis, the single iteration will be divided into 3 stages:

1. Time taken to write the image to shared memory buffer.
2. Time taken for the parallel hardware to complete calculations.
3. Time taken to read the results from shared memory.

Measuring the execution time will allow to calculate possible execution framerate (FPS) for an image stream. Measuring energy consumption in millijoules (mJ) will allow to compare energy efficiency of each device. Few different image sizes were selected to see what framerates are feasible with each platform (see figure 4). The performance is compared against a single core of ARM Cortex-A17 CPU, which would be a typical implementation of the *Sobel* convolution filter without any additional hardware acceleration.

The implemented algorithm was intended to be able to process image of any given size. Keeping that in mind and the fact that Epiphany co-processor's single core has only 32 kB of available local memory, the use of external shared DDR memory block was implemented. The program writes all image into shared memory block, then executes each of the parallel computing cores.

1.3. Measurement of energy consumption

We are going to analyze power consumption by the platforms during execution of the implemented algorithm. Current / voltage sensor INA219 [2] is used to measure power at 1 kHz rate. The sensor setup (see figure 5) was made to avoid power measurement influence possible by the additional electronics. The data is collected using microcontroller unit, which transmits measurements to laptop where they are recorded. Laptop uses wired LAN connection to receive messages from the platform performing calculations to capture beginning and end of the computation process. Energy equation $E = \int_0^t P(t)\mathrm{d}t$ derived from power equation in (Serway, 2013) is used to calculate the amount of energy used to process each frame. $P(t)$ is the measured power in Watts on time t. To check the reliability of the measured data Shapiro-Wilk test (Shapiro, 1965) will be used on calculated energy values. The statistical p-value threshold of 0.05 will be used to check the *null* hypotheses that measured data is of normal distribution. To provide additional confidence on measured data, Student's t-test will be performed to prove that mean values from both experiments has significant differences.

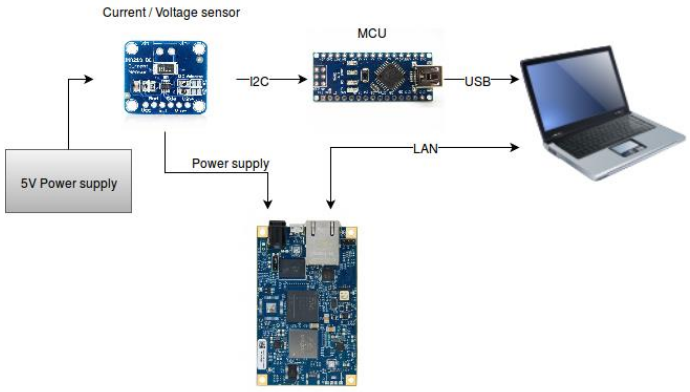

Fig. 5. Setup of power measurement sensor.

2. Experimental results

The *Sobel* filter was implemented on both OpenCL and eSDK frameworks for the Epiphany processor and a brief comparison of results was made. The main difference between these implementations is that eSDK required manual implementation of input / output image buffers, while OpenCL framework does manages buffers by itself. Figure 6 shows that eSDK framework has done the same calculations at least 3 times faster. The reason might be that the OpenCL is poorly implemented for the Epiphany processor and has too much overhead. Further experiments will be done using only eSDK, since it shows better results without doubt. The result values below are average values of 100 iteration processing the same image.

[2] Texas Instruments INA219 Current sensor, data sheet available online: http://www.ti.com/lit/ds/sbos448f/sbos448f.pdf

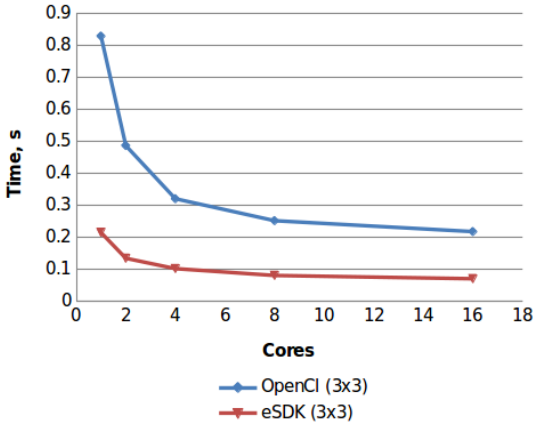

Fig. 6. Time taken to process a single frame using OpenCL and eSDK on the *Parallella* platform.

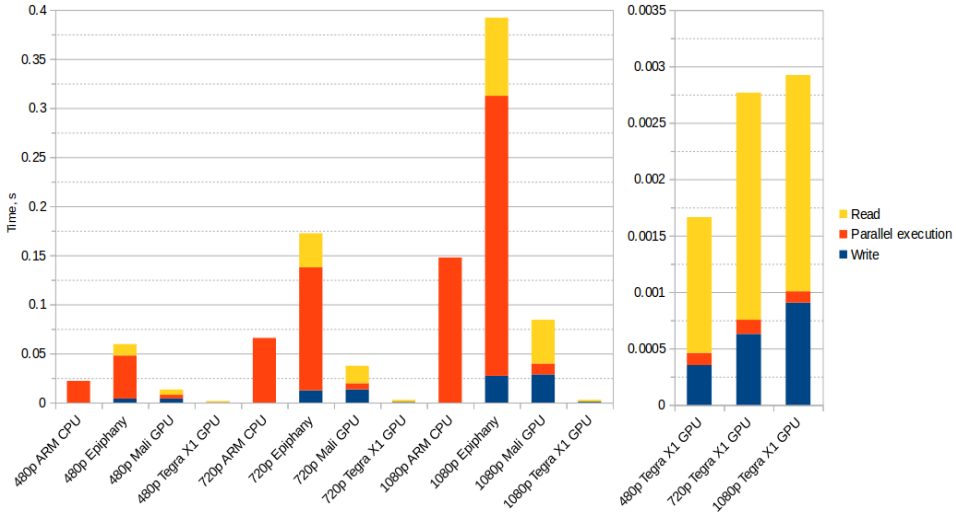

Fig. 7. Image processing performance.

Figure 7 shows the execution time taken to process the images of different resolution (480p, 720p and 1080p) with the two different platforms. The algorithm implementation on Radxa Rock2 platform using Mali GPU and OpenCL is at least 4 faster then implementation using Epiphany platform framework. The ARM CPU application is non-parallel and uses single Cortex-A17 core during runtime.

3. Energy efficiency

Histograms displayed in figures 9 and 10 shows consumed energy values for each of the experiment iterations, the red dashed line represents average value of all measured values. Table 1 presents average measured energy consumption for each experiment. The results show that by using Mali GPU T764 (on *Radxa Rock2* platform) may reduce

energy consumption 6.85 times comparing with 16 core Epiphany co-processor (on *Parallella* platform) and consumes 84.2 times less energy when using Nvidia Tegra X1 (*Airvision Core X1* platform).

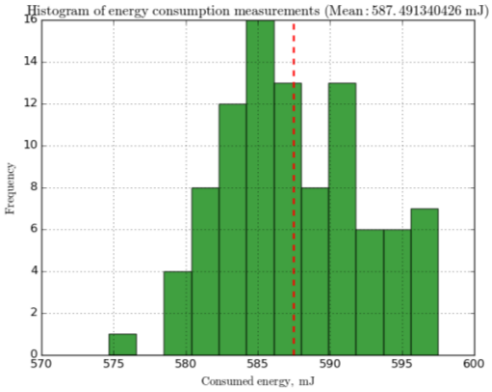

Fig. 8. Consumed energy histogram on *ARM Cortex A9 CPU*.

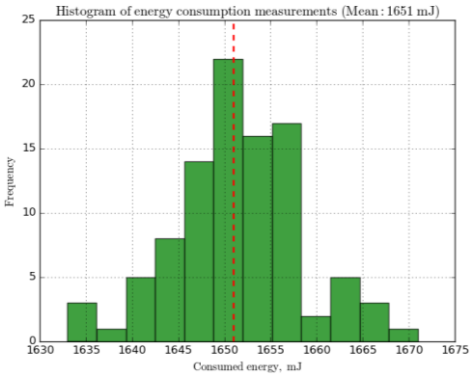

Fig. 9. Consumed energy histogram on *Parallella*.

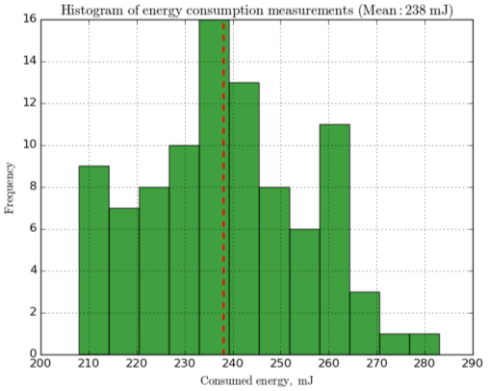

Fig. 10. Consumed energy histogram on *Radxa Rock2*.

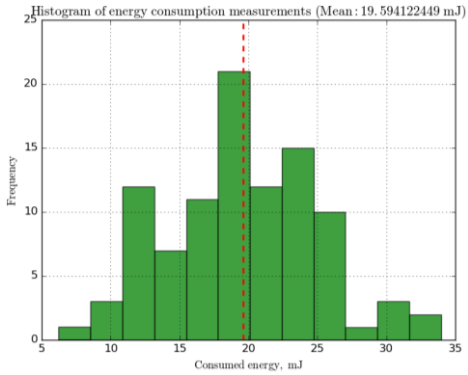

Fig. 11. Consumed energy histogram on *Airvision Core X1*.

Table 1. Measured average energy consumption per computed frame.

Platform	Energy, mJ
ARM Single core	587
Parallella	1651
Radxa Rock2	238
Airvision Core X1	19.6

Table 2. Shapiro-Wilk test results on measured consumed energy values

Data set name	W	p-value
ARM Single core	0.979	0.144
Airvision Core X1	0.984	0.153
Parallella	0.984	0.286
Radxa Rock2	0.980	0.165

Table 2 presents Shapiro-Wilk test results on measured data, all probability values are above 0.05, so we can state, that all measured data sets are normally distributed with 95% probability. Table 3 presents Student's t-test results between pairs of all data sets, so we can state that means between data sets are not equal with 95% probability (all p values are less than 0.05).

Table 3. Student's t-test results across data sets

	ARM Single core		Parallella		Radxa Rock2		Airvision Core X1	
Statistic	t	p	t	p	t	p	t	p
ARM Single core			-1207	0.0	187	0.0	752	0.0
Parallella	1207	0.0			742	0.0	1801	0.0
Radxa Rock2	-187	0.0	-742	0.0			119	0.0
Airvision Core X1	-752	0.0	-1801	0.0	-119	0.0		

4. Conclusions and Future work

In this paper we compared various heterogenous platforms and though *Parallella* platform has a new innovative co-processor technology it may be inefficient with input data intensive tasks, such as convolution filters (like benchmarked *Sobel* filter), with only 2.54 FPS processing speed on high resolution images. Since slow processing, the platform consumes 3x more energy than a single core ARM CPU.

Airvision Core X1 platform with Nvidia Tegra X1 processor showed the best results compared with any other platform, which may be expected, since it uses 256 cores for processing and other heterogenouos platforms has only 16 cores. It uses ~12x less energy for the same amount of calculations and is ~29x times faster than second best platform – *Radxa Rock2*, this is not only because execution is a lot faster because of the amount of cores, but memory transfer is ~26x faster.

Although *Airvision Core X1* platform is a almost6 times more expensive than other platforms (600$ versus 99$ and 120$ for *Parallella* and *Radxa Rock2* respectivly). To compare value-per-money metric we see that *Parallella* calculates 0,025 FPS/$ (minimum price per unit is 99$), *Radxa Rock2* can calculate 0.10 FPS/$ (minimum price per unit is 120$), *Airvision Core X1* 0.60 FPS/$ (minimum price per unit is 600$) so money-value is better with *Airvision Core X1* although *Radxa Rock2* is a cheaper solution.

Parallella platform is not able to surpass a single core application nor by processing speed (~3x slower) nor by energy consumption (~3x more energy). *Radxa Rock2* is ~5x faster and 7x more energy efficient then a single core application, while *Airvision Core X1* is ~142x times faster and ~84x more energy efficient.

Processing results on the *Parallella* platform is rather disappointing, the acceleration of processing is only 3 times faster using 16 parallel cores versus single-core implementation (on both eSDK and OpenCL). The reason may be the required data buffering due to insufficient memory on the co-processor. It is possible to avoid usage of the external memory buffer by writing image directly into the core's internal memory. This could improve performance, but constrains image size. Additional experiments would be required to backup this statement. Also, the FPGA system reviewed in the related work has shown potential in performance and power efficiency compared with

traditional platforms. Future work may include a comparison of currently benchmarked platforms and the FPGA.

Acknowledgment

The results of this research was obtained during development of project "Development of hybrid unmanned air vehicle for homeland defense purposes (No. KAM-01-08)" coordinated by Vilnius University.

References

Choi, S. (2003). Energy-efficient Signal Processing Using FPGAs. *Proceedings of the 2003 ACM/SIGDA Eleventh International Symposium on Field Programmable Gate Arrays*, 225-234. Monterey, California, USA: ACM.

Chowdhary, G. (2013). GPS-denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft. *Journal of Field Robotics*, 415-438.

Fowers, J. (2012). A Performance and Energy Comparison of FPGAs, GPUs, and Multicores for Sliding-window Applications. *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 47-56. Monterey, California, USA: ACM.

Grasso, I. (2014). Energy efficient HPC on embedded SoCs: Optimization techniques for Mali GPU. *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, 123-132. IEEE.

Knezovic, J. (2014). Are Your Passwords Safe: Energy-Efficient Bcrypt Cracking with Low-Cost Parallel Hardware. *WOOT'14 8th Usenix Workshop on Offensive Technologies Proceedings 23rd USENIX Security Symposium.*

Munshi, A. (2011). *OpenCL programming guide.* Pearson Education.

Nardi, L. (2014). Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM. *arXiv preprint arXiv:1410.2167*.

Olofsson, A. (2014). Epiphany, Kickstarting High-performance Energy-efficient Manycore Architectures with. *arXiv preprint arXiv:1412.5538*.

Pingle, K. K. (1969). Visual perception by a computer. *Automatic interpretation and classification of images*, 277-284.

Serway, R. (2013). *Physics for scientists and engineers with modern physics.* Cengage learning.

Shapiro, S. S. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 591-611.

Tompson, J. (2012). *An Introduction to the OpenCL Programming Model.* Digital version.

Uragun, B. (2011). Energy efficiency for unmanned aerial vehicles. *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, 316-320.

Varghese, A. (2014). Programming the Adapteva Epiphany 64-core Network-on-chip Coprocessor. *Parallel \& Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*, 984-992. IEEE.

WEB (a). *Airvision Core X1*. `http://www.airvision.io/core-x1`

WEB (b). *ARM Mali OpenCL SDK*. `http://malideveloper.arm.com/resources/sdks/mali-opencl-sdk/`

WEB (c). *Coprthr API reference*. Retrieved from `http://www.browndeertechnology.com/docs/coprthr_api_ref.pdf`

WEB (d). *piphany SDK Reference.* `http://adapteva.com/docs/epiphany_sdk_ref.pdf`

WEB (e). N*vidia CUDA Getting Started Guide For Linux.* Retrieved 03 05, 2016, from `http://developer.download.nvidia.com/compute/cuda/7_0/Prod/doc/CUDA_Getting_Started_Linux.pdf`

WEB (f). *Nvidia tegra x1 series processors data sheet.*
http://developer.download.nvidia.com/assets/embedded/secure/jetson/TX1/docs/TegraX1_Embedded_DataSheet_DS07224007v1.0.pdf?autho=1457984395_869543f9d0aec53c36939bf4e9acdb4c&file=TegraX1_Embedded_DataSheet_DS07224007v1.0.pdf
WEB (g). P*arallella - 1.x Reference Manual.*
http://www.parallella.org/docs/parallella_manual.pdf

Received March 18, 2016, accepted March 23, 2016