DSD-i1: A mixed functionality development board geared towards digital systems design education

Anastasios Fanariotis
Digital Systems and Media Computing Laboratory
Hellenic Open University
Patras, Greece
a.fanariotis@eap.gr

Theofanis Orphanoudakis
Digital Systems and Media Computing Laboratory
Hellenic Open University
Patras, Greece
fanis@eap.gr

Vasilios Fotopoulos
Digital Systems and Media Computing Laboratory
Hellenic Open University
Patras, Greece
vfotop1@eap.gr

Paris Kitsos
Electrical & Computer Engineering Department
University of Peloponnese
Patras, Greece
pkitsos@ieee.org

***Abstract*—The present paper describes the design and implementation of a development board, designed in Digital Systems and Media Computing Laboratory of the Hellenic Open University which is very active in the field of digital systems design. The board hosts an MCU and an FPGA on the same PCB, cooperating with tight interconnection between them and supported by a set of basic on-board peripherals that cover some of the most essential educational examples in the field, minimizing the need for external devices. The design is geared i) towards ease of use in order to alleviate any initial setup stress by students or inexperienced designers and ii) towards low-cost fabrication in order to facilitate educational institutions who provide distance education to offer this board to every student for out-of-laboratory usage.**

The design process, consists of three main stages; the PCB design, the firmware development and the lastly the host computer software development. The architectural and design choices made for each stage are described fully later on the paper with each decision balancing between ease-of-use, cost and functionality, in the form of offered services.

The board functions as a very low-cost laboratory educational platform for both low level HDL training as well as higher level MCU Firmware programming, supporting even more complex scenarios of FPGA softcore usage and programming or concurrent usage of FPGA and MCU in complete System-on-Chip (SoC) designs.

I. INTRODUCTION

The educational value and contribution of laboratory exercises and experiments in the field of science is well documented and widely acknowledged. Particularly in the area of digital systems design and at any level or educational subject, such as hardware or software development, such training modules can be deemed as necessary, because they present the opportunity to the student to apply any theoretical knowledge to experiments and directly observe the results. In this way, real world hands-on experience and deeper understanding of each educational subject are gained. Achieving this kind of functionality, requires the use of laboratory equipment and in many cases software toolchains in order to construct the experimental array. So far, in the field of digital design, this task was accomplished by interconnecting basic integrated circuits in Dual in-line packaging (DIP) like gate integrated circuits (ICs) or low-end microcontrollers (MCUs) on a breadboard. However, the recent developments over the last decade in the field of information technology and particularly in the areas of digital design, software development and system implementation have dramatically changed the requirements and therefore the educational effectiveness of using solely such type of hardware in the experimental arrays of a laboratory. Digital designs are now implemented using hardware description languages (HDLs) in field programmable gate arrays (FPGAs) instead of the previously used integrated gate circuits, plus there are no modern microcontrollers available that can easily be used in their primary form/package because of their complexity and lack of DIP packaging.

This problem was partially solved with the "arrival" of the first development boards in the form of microcontroller driven systems like Arduino and FPGA-powered boards such as MOJO. These systems gave the opportunity for complex systems design with modern development tools albeit bearing an additional financial burden compared to the simple gate ICs or single low-end microcontrollers. This was the result of their increased cost, as well as, the need for different boards for each educational subject. Other than this, the introduction of modern design tools increased the complexity for the inexperienced users presenting them with a "steep" learning curve especially in the area of initial system setup, while in other occasions easy-to-use tools like the Arduino IDE completely hid the MCU initialization options from the end user, making it unsuitable for advanced users or advanced learning subjects. It was also noticed that many of the commercially available development boards demanded additional programming hardware (e.g. JTAG), making the financial burden even heavier.

In this paper the possibility to minimize cost and increase flexibility driven by means of designing a single, mixed development board is explored. The requirements include support for modern microcontroller and FPGA technologies for both educational or experimental purposes, as well as support for a multitude of software development tools for assisting users with different experience levels according to the learning requirements or complexity.

The designed system is based and therefore is, a fully open system on both hardware and software aspects, in order to encourage usage and lower the cost. A prototype was built and the first spinoff of the design was used on a real digital systems design course (Hellenic Open University – Embedded systems design and microcontroller applications for the internet of things) with a broad learning area ranging from digital system design using VHDL, to microcontroller firmware development for IoT applications. From this course

information about the system's functionality and effectiveness was extracted by asking the students to fill-in a questionnaire.

The rest of the paper is organized in five additional sections, starting in section II, with the related work and current commercial availability of similar systems. In section III the system architecture is described among the different options and design choices made. In section IV the system implementation and usage is presented, while in section V an evaluation is attempted according to the questionnaire results and other observations made during its usage period.

Finally, all conclusions are drawn in section VI accompanied by a group of proposals for future work in order to increase the system's functionality and usability.

II. Related Work

There are quite a lot of development boards commercially available nowadays. However, most of them are designed towards a single application area and with a certain method of development in mind; thus, they are not so well suited for use on a wide variety of educational topics on the digital systems curricula. One can cover most aspects on this area by using multiple boards, with each one used on a different educational subject. This however creates additional cost and increases the difficulty of learning since each board is usually complemented with different software tools and has a totally different functional description and hardware arrangement.

This section presents a report of the current state of the art in the area previously described for both MCU and FPGA development boards. As state-of-the-art, in this report, are considered some well-known and widely accepted systems by the community, be it simple users, electronic or software engineering practitioners or educational institutions and individuals. The first example is Arduino [1]. This is a development board based on open hardware and software that set the foundation for every open, easy to use development board designed hereafter. The Arduino today is in fact a full-blown development platform with a multitude of hardware to choose from, ranging from 8-bit AVR MCU boards, up to 32-bit ARM Cortex driven alternatives. It is supported by its own integrated development environment (IDE) and focuses towards ease-of-use for inexperienced users [2]. Arduino has been exceptionally embraced by both the educational community [3] and simple makers, and it does excel in quickly introducing students into the digital systems world. The Arduino makes use of bootloader technology in order to load user firmware into the MCU, thus eliminating the need for extra and usually expensive programming/debugging hardware; this lowers the cost and increases the ease of use. On the other hand, that being said, it does hide most complexity of the MCU peripheral initialization or functionality, making it rather unsuitable for advanced learning subjects and while one can use an external JTAG programmer and a suitable IDE to reveal all the options of the MCU for advanced development, such functionality is not natively supported and may overwrite the bootloader making the board unusable by the Arduino IDE.

Another issue is the MCU technology. Arduino does come with a broad selection of hardware but is mostly known and accepted as Arduino UNO driven by a -now aged- 8-bit AVR MCU. It is an excellent MCU for initial learning purposes but lacks the resources for implementation of more advanced designs. On the other hand, the ARM based Arduinos are not so widely accepted and they are voltage incompatible with most of the additional hardware offered in the form of "piggyback" devices called shields.

The next system examined here is called LeafLabs Maple [4] and it was the first attempt of designing an Arduino-like fully open board with a modern microcontroller more than three years before the existence of the first official 32bit capable Arduino (the arduino DUE). It was based on an ARM-Cortex M3 MCU, used bootloading technology in order to achieve low cost and ease of use and implemented a modified Arduino IDE with extensive MCU peripheral library support as a development environment. The project was not so widely accepted in the beginning mainly because of the voltage incompatibility with the then-available peripheral breakout modules and shields and was abandoned in 2015, but it gave birth to a new, purely community driven and supported project, called STM32Duino [5]. This project was based on an extremely low-cost Maple mini variant, named by the community as "blue pill" and today rivals the Arduino in ecosystem size.

Both projects eliminated resource constrictions for advanced designs and while the usage of Arduino IDE kept the true flexibility of the MCU hidden from the end user, the first projects that made use of both the Arduino-like Hardware Abstraction Layer (HAL) and register-based initialization and usage start to emerge and are embraced by the community. In fact, STM32Duino initiated the convergence of using a high-level HAL to using a low-level register approach of development methods. Apparently, the need for more advanced projects and the available capabilities of the MCU drove the end-users into seeking the most effective method of development with the side-effect of opening the way for usage in wider area of educational subjects. In order to achieve a better grasp of the differences of these boards a comparison table has been created and presented bellow (Table 1)

Table 1 - Arduino vs Blue Pill

	Arduino UNO	**Blue Pill**
MCU Architecture	AVR 8 bit	ARM 32 bit
MCU Core Clock	16MHz	72MHz
MCU available Storage	32KB	64KB
MCU available SRAM	2KB	20KB
Board Available I/Os	20	32
Voltage Level Spec	5V	3.3V
Hardware Timers available	2x8bit, 1x16bit	4 x 16bit x 4 Channels each, 1x24bit Systick
Hardware Functions	PTC, 1xADC 10bit	DMA, CRC, LIN, IrDA, 2xADC 12bit, NVIC
Interfaces	1xUSART, 2xSPI, 1xI2C	1xUSB, 1xCAN, 3xUSART, 2xSPI,
Current MSRP (2019)	20€	4€

The list of MCU based development boards with their descriptions can practically fill hundreds of pages, however these two examples are the most prominent in to establishing a good grasp of today's state-of-the-art level. Next in line of this report is the available FPGA-based development boards. Following the success of Arduino, a lot of developers rushed into designing such cost-effective boards thus bringing the HDLs closer to end users. Among the total commercially available boards there are a few examples of such systems that are widely accepted by the users and are following different design methods in order to increase usability. The first one is the Papilio Pro [6], a Xilinx Spartan 6 based development board with an Arduino compatible pin headers arrangement, 9K LEs and 64Mbits of external RAM available. In order to decrease the cost this board eliminates any proprietary programming JTAG hardware by using the same integrated circuit for both user connectivity and bitstream uploading. This in turn gives rise to the need for a software package that the end-user must use in order to upload the bitstream design to the FPGA IC. The result is that the user is not able to upload his or her design directly from the EDA environment witch in this occasion is the Xilinx ISE. This method does increase complexity but not significantly, after all, it eliminates the need for setup of the external JTAG hardware to the IDE which in many cases is proven to be cumbersome for inexperienced users. The most prominent problem with this board in educational or otherwise usage is the fact that Xilinx dropped the support for ISE in order to force migration to its new EDA called *vivado* and the Spartan 6 series is not supported by the new EDA.

The next board under investigation is the Terasic DE0-nano. An extremely feature rich, Intel Cyclone IV based system. The usability design of this board is completely different since it implements onboard the Intel's official JTAG hardware "Blaster Cable" giving the opportunity to the end user or the student to directly upload the HDL design from within the Quartus II EDA. Other than that, this board comes loaded with a significant number of external peripherals such as accelerometers, LEDs, tactile interfaces and A/D converters alleviating the need for usage of external breakout modules but simultaneously increasing the cost. The FPGA IC offers 22K LEs deemed to be more than enough for educational projects and the DE series development boards are well known in the learning community as laboratory experiment platforms or remote Lab applications [7] thus its educational effectiveness is proven albeit with a cost that does not permit the offering of this board to every student for home-usage. The main specifications for each board are presented bellow (Table 2)

Table 2 - Papilio Pro vs De0-nano

	Papilio Pro	**DE0-nano**
Available LEs	9K	22K
Available BRAM	576Kbits	594Kbits
Available Board I/Os	48	96
HOST Connectivity	1xUSB	1xUSB
Board Peripherals	1x64Mbit SDRAM, 1x64Mbit SPI Flash	1x256kbit SDRAM, 1x EPCS 64Mbit, 1x2Kbit i2c EEProm, 1xAccelerometer IC, 1x ADC IC
User Interaction	1x Reset button, 1xLED	8xLED, 2xTactile buttons, 4xDip switches
Programming Hardware	FTDI Mode, no Debug	Integrated USB Blaster Cable
MSRP	~66€	~88€

Until now it must have been made apparent that there is a gap between the Low-Level Hardware Design (FPGAs) and High-Level Development (MCUs) hardware-wise. In order to cover a wide area of digital design learning or development two or more boards are needed as development platforms and if the need for a combined design is arisen, then the complexity of interconnection between them will make the undertake much more difficult and maybe, extremely time-consuming diverting the student from the real educational subject.

This gap was filled by the appearance of the so-called mixed development boards that implemented a microcontroller and an FPGA on the same Printed circuit board (PCB). While there are FPGAs available with integrated MCUs into their fabric the combination of HDL and Assembly or C development on these devices demands expertise that is well beyond the generalized educational aspect. A mixed board with discrete MCU and FPGA ICs is a much more well-suited platform for educational usage on simple to medium complexity designs. Currently there are not too many options for this type of development boards, at least, in the form of open hardware/software or within a logical price range.

The first selected board is the MOJO V3 [8] that is designed around an 8bit AVR MCU, similar to that of the Arduino and a Xilinx Spartan 6 FPGA and so it inherits most of the functionality offered by Arduino UNO and Papilio Pro but in a single board form. The MCU can be programmed by means of a bootloader and supports the Arduino IDE after installing the correct board package. An interesting design implementation by the creator of this board is that the MCU is used firstly as an FPGA bitstream upload programmer eliminating the need for extra JTAG hardware thus lowering the cost and secondly as an ADC converter peripheral for the FPGA. Sadly, most of the MCU pins are connected solely to FPGA I/Os and are not offered as I/Os through a header, making this board mainly functional for purely HDL or interconnected mixed designs. There is no way that a user can get access to MCU I/Os (other than those that are set-up for ADC) without implementing a passthrough HDL design to FPGA. It uses Xilinx ISE as a main HDL development environment, and the Arduino IDE for any desired MCU functionality plus an extra proprietary IDE that the user may use to develop Lucid HDL designs and upload them to the FPGA. Other than these, the 8 LEDs present on the PCB are extremely useful for directly implementing simple HDL designs of educational projects for beginners.

The last board examined in this section belongs to the Arduino ecosystem and in named "Arduino MKR Vidor 4000". This is a very small factor board (similar in size to the blue pill board) that is designed around an Intel Cyclone 10 FPGA and an ARM Cortex M0+ microcontroller. The choice of these two ICs is not random since they are both capable of low-power functionality, making the Vidor the first available

mixed board geared towards low-power applications. An interesting design aspect is the system's support of using a LiPo battery as a power source and the integrated PCIe breakout connector at the edge of the board making this board both portable with low power capabilities and usable as an extension card on a HOST computer PCIe slot. Other than these the board comes preinstalled with a Wifi/bluetooth capable radio module. Unfortunately, while hardware-wise this board is extremely reach and attractive, especially for IoT applications, when it comes to development options it only supports Arduino IDE and uses the FPGA device as a peripheral through a C++ library. Although direct FPGA programming has been promised by the creator nothing has been implemented to this day. This board is geared towards interconnected mixed designs or microcontroller-based designs only and even in the case of mixed projects there is no strong/fast interconnection available between the MCU and FPGA.

This section will be closed with a comparative table (Table 3) and the initial conclusion of the current state in this field that there is no generic board available to cover a wide range of applications, educational or otherwise. Every board inexplicably limits the developer into choosing a board that either mostly supports microcontroller designs or FPGA designs with very little support for balance between them. There is really no credible reason for the lack of bitstream uploading function or why a board header or headers cannot be shared between MCU and FPGA I/Os, enriching the interconnectivity of the ICs and releasing the full flexibility of these modern devices to the designer. There are of course other options available, either the use of discrete boards or the use of a much more expensive "formal" development board, with an increased financial or time cost.

Table 3 - Mojo-V3 vs Arduino Vidor 4000

	MOJO V3	**Arduino Vidor**
LEs	9K	16K
BRAM	576Kbits	504Kbits
FPGA Programming	MCU driven	Not available / through C coding
MCU programming	Bootloader	Bootloader
MCU I/O Availability	ADC Only (8 pins)	22 Pins
FPGA I/O availability	Full (84 pins)	shared pins (21)
Supported IDEs	Xilinx ISE, Arduino	Arduino
HOST Connectivity	1xUSB	1xUSB
Board Peripherals	8xLED, 1x Tactile button, 1x4Mbit SPI flash	64Mbit SDRAM, 16Mbit Flash, BT, WiFi, HDMI out.
MSRP	66€	63€

III. DSD-i1 Architecture

Since the requirement for a more flexible board has been established in the previous sections, the choice to design a development board that will alleviate any restrictions posed by the currently commercially available boards is well justified as well. The design specifications of this board are described in this section. Such an endeavor consists of multiple design steps starting with the system architecture definition hardware and software wise. In this section the system -called from now-on DSD-i1:(**D**SMC **S**tudent **D**esign Board – **I**ntel variation **1**) - architecture is presented, along with the necessary justification for every design choice. The DSD-i1 design and architecture consists of 3 main subsystems. The PCB design which includes the hardware selection, the MCU firmware development that will be providing any onboard services and finally the Host Computer software package development, containing the graphical user interface, device drivers and any other support packages required.

For each subsystem, prior to any design step, a group of requirements are defined sourced by the observations made in section II.

A. Hardware Selection and PCB Design

The DSD-i1 should achieve a high level of flexibility, ease of use and cost effectiveness. MCU-wise in order to alleviate the need for external programming hardware a bootloader should be used, connectivity-wise the MCU should natively support USB connections and multiple peripheral interfaces. As for the architecture an ARM based core is extremely attractive as a modern MCU with plentiful resources. There is also the advantage of the experience gained over such systems versus the requirements of the job market since the ARM architecture is widely used in mobile and ubiquitous/ embedded computing systems. The selected MCU should also support a variety of development environments for both beginners and experienced developers. Under this scope a variant of the blue pill's MCU is defined as an excellent choice since other than the capabilities, architecture and resources of the Cortex M3 core there is a bootloader available as open source project and a huge community supporting different open sourced development environments like Arduino IDE and System Workbench. ST microelectronics also provides free of charge an initialization Utility called STM32CubeMX that supports a multitude of IDEs with KEIL MDK among them. That been said the STM32F103CBT6 is an identical MCU to the blue-pill's one that comes with 128KB of available flash storage. This MCU extends the possibility for large projects without any significant financial burden, retains the compatibility with the works of the community as an open source project, and provides the ability to support an extremely wide range of applications and levels of difficulty/complexity in educational or otherwise projects such as RTOS driven projects [9].

The choice of FPGA IC should cover the subjects of EDA support/compatibility, LEs availability according to educational needs and cost effectiveness. Since Xilinx has dropped the support for ISE, any Spartan 6 FPGA is non-selectable. Any newer Xilinx FPGA comes with a significant increase in cost and Xilinx Vivado EDA does not currently support a top-level design in schematic form and thus deemed unsuitable for beginners. Intel on the other hand offers competitive prices on rather large FPGAs and continuing support of Quartus II EDA. Lattice is better priced but geared mostly towards small FPGAs. The major requirement here is the support for SoftCores and the availability of such designs in easy to use free of charge form. Intel provides Quartus II Lite with NIOS II-e[10] Softcore free of charge, it also offers Cyclone IV FPGAs in extremely low prices with their entry

level 6K LEs EP4CE6 FPGA capable of implementing the NIOS II softcore with 20K SRAM leaving enough free space for complementary small designs if needed. This FPGA IC although it requires an additional power resource compared to the Xilinx Spartan 6 series, is exceptionally well-suited for DSD-i1, plus the next model of this line EP4CE10 with 10K LEs is pin compatible with the smaller one thus giving the opportunity to choose between them during PCB assembly.

Peripheral devices are not always easy to choose, especially when it comes down to cost effectiveness. Since on-board LEDs are extremely useful as a direct optical communication device with the user and are very low priced, their inclusion is an easy design choice. Keeping in mind typical educational examples of computer architecture and simple register functions (e.g arithmetic functions between two 8-bit registers) a multitude of 24 LEDs, arranged in three rows of eight LEDs each, will give the opportunity to the students to visually observe the step by step functionality of a designed ALU. An extra LED should be added common to the MCU and FPGA as Arduino IDE requires it and any FPGA design may need an extra indication of illegal states like overflow etc. This sums up to 25 LEDs total to the board.

Onboard Storage and extensibility will be implemented by adding an SPI flash EEPROM IC that should be available to both FPGA and MCU. The worst-case scenario is to be able to fit at least one bitstream file for boot-up reprogramming of the FPGA. EP4CE6 and CE10 FPGAs require 2944088 bits of space for an uncompressed raw binary format file, this size sets the requirement for at least 4Mbit size of flash in IC. However, the difference in price between 4Mbit and 8Mbit SPI flash ICs is negligible so it is prudent to opt the larger storage in order to extend the board's capabilities and increase the lifespan of the Flash memory by decreasing the write/erase cycles on the same address space. This design issue does not end here, there is the possibility of IoT or other embedded application implementations that may require much more storage. The solution to this problem is the inclusion of an SD card slot connected over a single SPI channel and implemented as 3.3V first row only SD/non-UHS for compatibility reasons.

Connectivity-wise, two pin headers of 2x40 pins each with the "standard" 2.54mm pitch are deemed enough for any application that needs external modules connectivity. All MCU pins should be available on one row and numbering should be compatible with STM32Duino applications in order to directly support Arduino IDE. These pins can be common between FPGA and MCU with the exception of ADC-input pins in order to avoid any signal interference by the FPGA's output stage weak pull-up. Any common pins should be protected from short-circuit between FPGA and MCU (e.g. a single pin is driven high by the MCU and the same pin LOW by the FPGA because of user-error). Other than these, there should be strong interconnection between MCU and FPGA and the capability for the MCU to function as a bitstream loader to FPGA in Serial Passive mode. This means that as many as possible parallel and serial communication channels should be common between the FPGA and MCU IC making it strongly interconnected. While this poses no problem for the FPGA IC, since any physical trace/pin may be designed/ programmed for virtually any functionality, a careful design must be considered from the MCU side because each physical pin has predefined functionality.

Finally, since the selected MCU is capable of natively supporting USB connectivity, a single USB connector should be added as a micro-usb package because of the extreme availability and low cost of such USB cables. The USB port will be used as a data connection between the host PC and the board, for bitstream uploading, MCU reprogramming and user-interaction (e.g. a serial console) and as a power source for the board. Because of the USB standard Power Limitations [11] and the capability of the FPGA to draw extreme amounts of energy, a current limiting circuit should be included in the design.

Although a four or more-layer PCB is the standard for such designs, the FPGA's multiple power sources make it possible, with careful design to implement this in dual layer, keeping in mind that signal routing requirements are geared towards simple educational experiments. Furthermore, to decrease the assembly cost, all components should be placed on one layer.

Since this is an open hardware project, the design should be easily available to the users, so a wide-spread CAD utility like eagle CAD was used for the design. The basic block design and the eagle cad PCB design are shown in the images bellow (Figure 1,2).

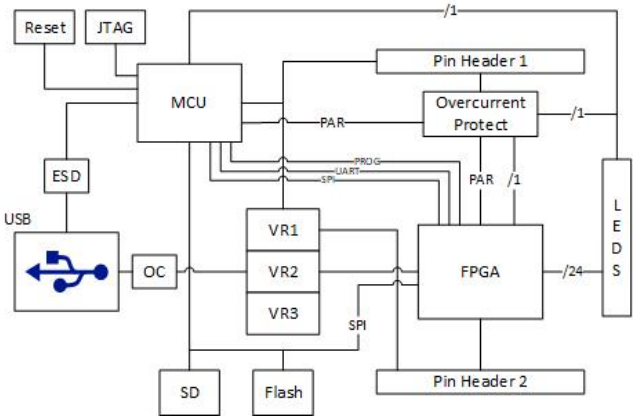


Figure 1 - DSD-i1 Architecture Block Diagram

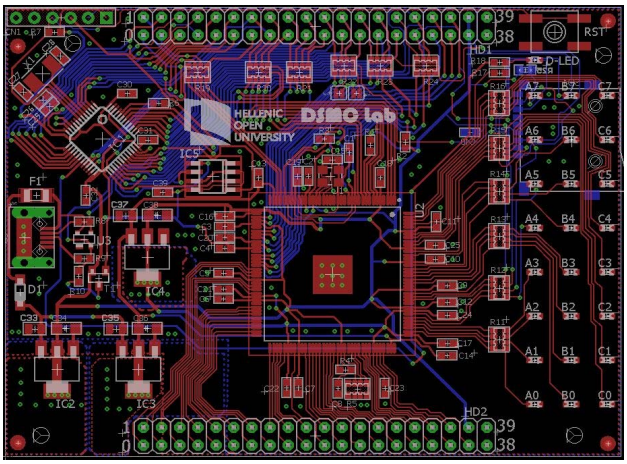


Figure 2 - DSD-i1 Eagle PCB design

B. MCU Firmware Development

According to the Functionality and open-source requirements the MCU firmware should contain a bootloader and a main program that will enable FPGA bitstream Loading, NIOS HEX uploading and Serial post communication with the FPGA. The bootloader used is based on the Blue Pill bootloader [12] with slight modifications to wait and reset timing signals for stability reasons. In order to achieve all the required functionality, a simple communications protocol was developed that consists of four main simple commands to switch the functionality between these services plus an initial wait-state. All services are attached to a state machine structure with dynamic timeout returning to the initial state in order to stabilize the functionality in case of failed transmissions. In order to discriminate between FPGA and MCU serial console communications, a sub-protocol was developed that uses plain serial comms for the MCU and 2-byte communications for the FPGA in the form of header-payload. The header is created by the GUI application and instructs the MCU to filter the header and pass the payload to the FPGA. The pure payload is transferred to the FPGA making its implementation transparent to the end user, as long as the offered serial console software is used (described in the next section). Since the header consists of a full byte the protocol is extensible and may support more control functions if needed. Finally, an extra service was developed that translates the Intel's RBF format to pure binary programming payload for the FPGA. It is useful to point here that no bitstream compression is supported since, as it is unnecessary because of the bitstream's small size.

Since this is an Open Source project and availability of the code is critical for all users, the firmware was developed in Arduino IDE. This choice makes it easy for all non-experienced users to modify it, although some modification to the basic libraries was made in order to support higher bitrates and DMA capability for SPI communications to increase transfer rate of bitstream to FPGA. Up to 36Mbps (36Mhz SPI clock) operation was proven to function correctly for FPGA Design uploading although the signal paths on the PCB are not designed for such high speeds other than the provision for minimizing their length.

C. Host Computer Software Development

The host computer software package is the main point of interfacing the DSD-i1 board to the user. While the board is directly compatible with the Arduino IDE after the blue-pill's support package is installed, to be able to fully take advantage of the functionality the main GUI utility should be used on every other type of development. The utility is called X2Loader and is an easy to use GUI that access the board's resources in an intuitive manner. This utility supports three main functions. First, it can undertake the uploading of RBF compiled HDL designs. Such designs can be created using the Quartus II EDA by selecting output compilation files as uncompressed RBF (Raw Binary Format). Second, it can undertake the uploading of firmware compiled in Intel HEX format to a NIOS soft-core implementation on the FPGA. In this case the design can be created by using the Quartus II EDA SBT tools, after the implementation of an SPI to Avalon bridge connection on softcore design (Qsys) that serves as an SPI to memory access point, which allows for direct access of the softcore's SRAM by the MCU. Third, it takes care of any serial communication functions between FPGA, MCU and user console. It is useful to point here that the serial console can discriminate between messages coming from the FPGA and messages coming from the MCU and adds a string message in front of every such message indicating the source. However, in the reverse direction, i.e. in the case of message transmission towards the DSD-i1 board, the user must choose where the message will be sent by selecting or deselecting the "ARM console" checkbox (Figure 3). Finally, an extra secondary function named "board recovery" added in order to increase ease of use in case of problems or user errors that may impair the board's functionality (soft-errors). This function loads the default firmware to the board by instructing the user to follow a step-by-step procedure.

Complementary to this utility another software sub-package was created in order to support MCU programming from more flexible IDEs. The IDE chosen to be supported was KEIL-MDK that is in fact the most widespread professional development tool for ARM microcontrollers and if offered free with a compiled code size limit of 23KBytes. The utility named "KEIL Loader" was developed to run from within the KEIL IDE and uses the MCU's bootloader for uploading the firmware to MCU without the need for external JTAG hardware (Figure 4). For debugging purposes, the developer may use the *"sprint"* command in order to print debug messages to the x2loader console. That been said there are a few facts that need to be stated here. Firstly, KEIL is actively supporting educational institutions by offering free professional licenses with no code limitations and secondly it is extremely easy to use the same tool to support other IDEs like the Open Source System Workbench of ST microelectronics or the Atolic TrueSTUDIO recently obtained by ST microelectronics and offered free of charge to users of STM32 MCUs. All these IDEs may be used in conjunction with the STM32CubeMX utility that is a graphical initialization code generator offered free by ST Microelectronics.

All the software packages listed above were developed using the JAVA programming language in order to achieve multiplatform functionality if such requirement arises in the future.

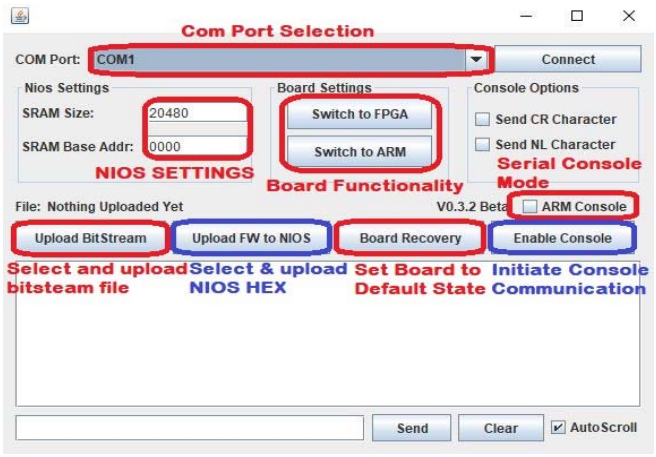


Figure 3 - X2Loader GUI

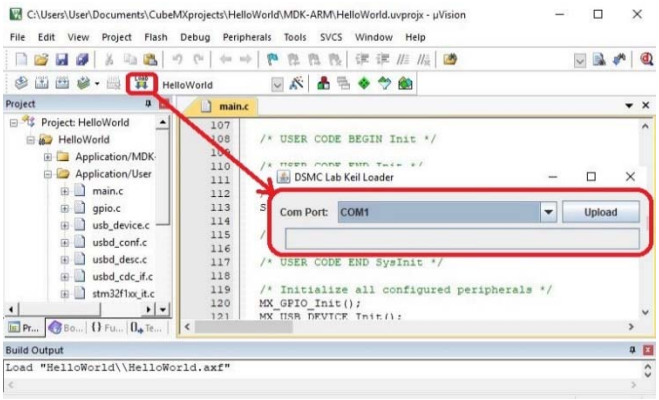


Figure 4 - KEIL Loader Utility

IV. System Implementation And Use

In order to increase ease of use, reduce the initial setup time and minimize any possibility of errors from the end-users every software package developed plus the device drivers for the USB connection was placed inside an installer package. The installer package places the software package files to predefined locations thus all users may use the software's documentation in order to further setup or troubleshoot the installation should any problems arise. The installer also checks if JAVA runtime environment is installed and instructs the user accordingly to download and install it.

After the initial installation the user may connect the board to the host computer. The board should be detected as a Serial Port Device or a DFU device in the operating system, in any case the board is ready to be used from Arduino IDE and X2Loader (interfacing to Quartus II EDA) without any further action or from the KEIL-MDK and STM32CubeMX with a few additional settings to the IDE. The final prototype of the board is presented bellow (Figure 5).

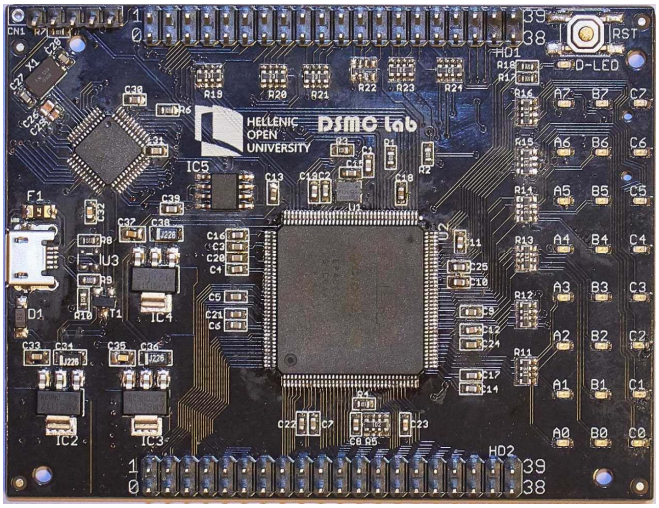


Figure 5 - DSD-i1 Prototype Board

The low quantity components price for this board plus the PCB price are given in the following table (Table 4). The price does not include the SD card slot on the bottom layer and the assembly cost since it was done in-house. The price is also affected by the availability of components during the prototyping phase.

Table 4 – DSD-i1 Cost

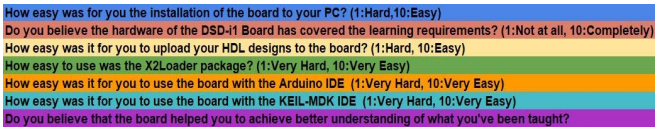
Device	Function	Total Price (€)
STM32F103CBT6	MCU	4,71
EP4CE6E22C8N	FPGA	10,16
LD1117S33TR	3.3 Vreg	0,173
LD1117S12TR	1.2 Vreg	0,207
LD1117S25CTR	2.5 Vreg	0,19
ASEM1-50.000MHZ-LC-T	OSC	0,98
LFXTAL003210Reel	XTAL 8MHz	0,78
MBR0520LT1G	SC Diode	0,079
USB-Micro Connector	1 x USB Con	1,3
Capacitors	Various Capac	1,32
Resistor Arrays	Resistor Array	0,06
Resistors 0603	Various Res	0,117
LEDs	0603 SMT	1,375
Tact Switch	LED	0,162
HEADERS	HEADERS	3
POLYSWITCH	PDTA114ET	0,25
PRTR5V0U2X	ESD suppressor	0,217
AT25SF081	SPI FLASH	0,337
PDTA114ET	Enum Transistor	0,05
PCB	PCB	2
Total Cost		**27,46**

V. Evaluation

The system was used in a training course titled "Embedded design and microcontroller applications for the Internet of Things", offered by (removed for blind review) from which the evaluation result was sourced and are presented in this chapter. The evaluation was implemented via a questionnaire measuring the user effort during DSD-i1 usage for each development task, the ease of use of the GUI the completeness of the hardware and the effectiveness of the installation package. Finally, one question has targeted the educational effectiveness of the DSD-i1. All user questions requested a grading of 1 up to 10.

The installation package was found to be extremely useful by the users with an average of 9.6/100, Hardware completeness was graded with an average of 8/10, FPGA Bitstream Loading functionality effectiveness was graded with 7.9/10 and general X2Loader ease-of-use with 7.8/10. The Arduino IDE support was graded with a 9/10 and the KEIL Loader with an 8.2/10. The overall effectiveness of the DSD-i1 as an educational tool was graded with a 9.2/10.

All discrete services of the board scored high numbers, however there was another question on the questionnaire that set apart the users with prior knowledge of digital systems design from the beginners. The full results are presented below in charts (Figure 6, 7).



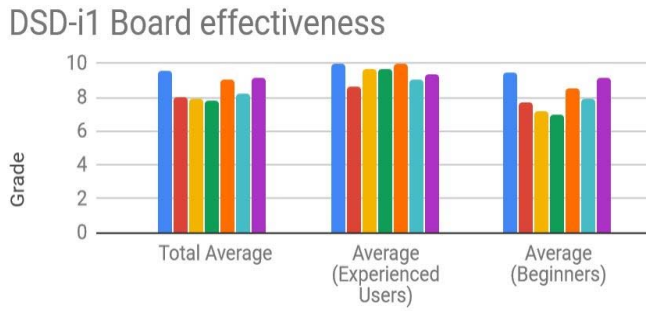


Figure 6 - Average results per question

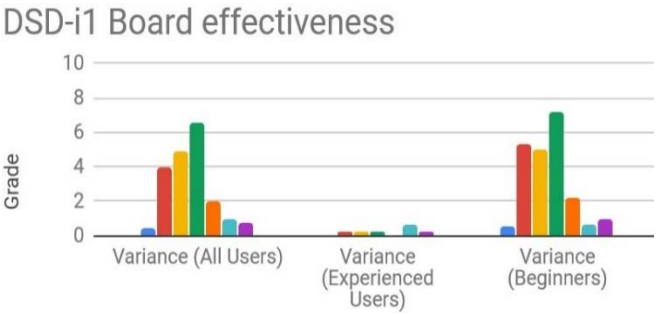


Figure 7 - Variance per question

There is a median chart available that "improves" the overall score; however, this is not the point here, variance between beginners and experienced users is proving the initial assumption that ease of use is a major factor for newcomers in the area of digital systems design. Initial hardware setup and familiarization with the development board and (especially) development tools present a challenge for beginners and form an extremely "steep" learning curve. Analyzing this chart also reveals the reason for the success of Arduino with its extremely easy development environment and an initial setup of "connect-and-forget/PnP".

Another aspect of the results that came as a complete surprise was the grading of the KEIL-MDK IDE versus the Arduino IDE ease of use on the DSD-i1. They both scored similar numbers with the same variance between the experienced users and beginners. Empirically, KEIL-MDK is considered extremely difficult for beginners, however in this case, providing an easy to setup tool for firmware uploading and the use of STM32CubeMX apparently made things almost equally effective as using an Arduino IDE by easing up the initial learning curve.

The similarity of the results between HDL designs and firmware development applied on the same board indicates that DSD-i1 achieved its goal of covering a wide variety of digital systems design educational subjects.

VI. Conclusion And Future Work

In this paper the design and usage results of a general-use development board were presented. The DSD-i1 board was deemed as a successful design in its goals of covering an extended area of digital systems design and retain a low cost. Moreover, the results indicate that the board managed to ease up the initial problems typically faced by beginners.

When the students were asked for any improvement suggestions, their response was overwhelming. Some suggested the inclusion of a 7-segment display on the board while others asked for more development IDEs/platforms support. Future work may include the following, according to user suggestions:

- Support for the online ARM MBED IDE/compiler [13]. This may be achieved by means of a bootloader upgrade
- Support for access of all (25) LEDs from MCU, this may be intergraded by auto-uploading a passthrough HDL design to FPGA since 24 LEDs are connected directly on FPGA pins and there are not enough I/Os available to the MCU
- Support for wireless connectivity without any external hardware module
- Support for more tactile interfacing (buttons & switches) on the PCB as well as inclusion of a 7-segment LED

Other than the board and software upgrades, it is a firm belief that in order to further improve the board's effectiveness and thus the learning process, more research is needed with measurements from training courses.

The user feedback was extremely useful and revealed aspects for improvements that were not easy to define prior to the questionnaire results.

ACKNOWLEDGMENT

The present work was undertaken in the context of the "Multiservice cApable iNtelligent TransportatIon Systems (MANTIS) project" co□financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE - INNOVATE (project code:T1EDK-04612).

References

[1] Arduino, S. A. (2015). Arduino. *Arduino LLC*
[2] Kushner, D. (2011). The making of arduino. *IEEE spectrum*, *26*.
[3] Galadima, A. A. (2014, September). Arduino as a learning tool. In *Electronics, Computer and Computation (ICECCO), 2014 11th International Conference on* (pp. 1-4). IEEE.
[4] LeafLabs Maple (2019, April 11). Retrieved from https://www.leaflabs.com/maple
[5] Arduino for STM32 (2019, April 11). Retrieved from https://www.stm32duino.com/
[6] Papilio Pro (2019, April 11), Retrieved from http://papilio.cc/index.php?n=Papilio.PapilioPro
[7] (Removed for Blind Review)
[8] Alchitry MOJO V3, (2019, April 11), Retrieved from https://alchitry.com/products/mojo-v3
[9] He, N., & Huang, H. W. (2014). USE OF FreeRTOS IN TEACHING A REAL-TIME EMBEDDED SYSTEMS DESIGN COURSE. *The ASEE Computers in Education (CoED) Journal*, *5*(4), 18.
[10] Nios II Core Implementation Details (2019, October 19). Retrieved From https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2cpu_nii51015.pdf
[11] Specification - Rev, U. S. B. 1.0, Jan. 15, 1996. *Sec*, *9*(2), 184-185.
[12] Clark R., STM32Duino Bootloader (2019, March 3), retrieved from https://github.com/rogerclarkmelbourne/STM32duino-bootloader
[13] Mbed Boards (2019, October 11), Retrieved from https://os.mbed.com/platforms/