MIPRO 2011, May 23-27, 2011, Opatija, Croatia

Embedded System Design with Maintenance Consideration

Xiaozhou Meng, Benny Thörnberg, Najeem Lawal
Department of Information Technology and Media
Mid Sweden University, Sundsvall, Sweden
{xiaozhou.meng, benny.thornberg, najeem.lawal}@miun.se

Abstract - This paper deals with the problems of maintaining a long lifetime embedded system, including obsolescence, function change requirement or technology migration etc. The aim of the presented work is to analyze the maintainability of long lifetime embedded systems for different design technologies. FPGA platform solutions are proposed in order to ease the system maintenance. Different platform cases are evaluated by analyzing the essence of each case and the consequences of different risk scenarios during system maintenance. Finally, the conclusion is drawn that the FPGA platform with vendor and device independent soft IP is the best choice.

I. INTRODUCTION

Modern embedded system designer considers various aspects during the design process, including performance, cost, power etc. However, one issue is often missing from this list, namely maintainability. Due to the rapid development in electronic technology, obsolescence and upgrade are inevitable for most embedded systems which may create a variety of troubles when maintaining a long lifetime system.

It is often the case that only a part of the system is actually obsolete or requiring modification. Unfortunately, the replacement or modification is usually as difficult as designing an entire system because the system has been developed as a single entity, with much interdependence between its hardware and software [1].

According to Moore's law, the number of transistors on a chip doubles every 18 to 24 months: poor planning with regards to parts obsolescence causes companies to spend progressively more in order to deal with the effects of aging systems [2]. In the commercial markets, electronics components in consumer electronics such as PCs or portable applications, for instance, are updated very rapidly, while in automotive, avionics, military application etc., the desired lifetime for these systems is many times longer than the obsolescence cycle for the electronic components used in the systems. For avionics and defense applications, systems face obsolescence even before they enter into service (due to the long design, manufacturing and test cycles).

In addition to obsolescence, a system will require re-engineering if its requirements change over time or if it becomes necessary to change the specifications. The availability of newer and better architectures (processors, interconnections and interface blocks) can provide the motivation for a re-engineering of a product. In the commercial arena, manufacturers must re-engineer their products in order to provide the new features required by customers, to incorporate newer technologies and standards, or to reduce costs and increase value [3].

The aim of the work presented in this paper is to analyze the maintainability of long lifetime embedded system for different design technologies. An industrial controller area network (CAN) controller system is used as our experimental system, which is implemented using different design technologies generating four different cases. The essence of each case and the consequences associated with different risk scenarios during system maintenance are analyzed. The result of this case study confirms that the proposed system on FPGA in combination with soft IPs (intellectual property) will increase the maintainability of a long lifetime embedded system.

II. RELATED WORK

Hitt and Schmidt [4] have investigated and assessed the impact of technology obsolescence and following on from this work many solutions then proposed to minimize such an impact. For example, it is possible to purchase vast stocks of components and store them as replacements if a component becomes obsolete. However, this does not enable new technologies to be used and the stock component will be depleted at some stage. Audsley, Bate and Grigg [1] propose using portable code which allows compiled software to be executed on any platform without change thus reducing the cost of hardware obsolescence. Meyer et al. [5] presented a model to develop an obsolescence mitigation timeline for the entire life support of complex electronic or long lifetime systems. When a product's obsolescence is unavoidable, a forecast regarding its life cycle end time becomes important. Sandborn et al. [6] present a method for a data mining based approach in order to forecast electronic part obsolescence. FPGA is a flexible and less costly replacement solution for an obsolete component. Velazco, Anghel and Saleh [7] propose a solution that emulated the obsolete Motorola 6800 processor using FPGA, which is a good suggestion for maintaining a current system, not initially implemented on an FPGA platform. However, none of these research works analyzed the essence of the embedded platform and thus it was not possible to significantly reduce either the risk or cost by their methods during system maintenance.

III. Problem Issue for Maintenance

For an embedded system designer, the hardware components and development tools are provided by providers. The final system will be delivered to customers. It is possible for the maintenance risks to arrive from these three groups.

A. Provider

Obsolescence is a state of a product's lifecycle when it is no longer produced (i.e. no longer provided by the provider). The growing use of commercial off-the-shelf (COTS) components and equipment increases the risk of obsolescence. The reasons for obsolescence could be technological, market or environmental etc. It is possible to divide obsolescence into several types for embedded systems:

- *Peripheral interface obsolete.* The peripheral interface standard is developing. A new standard will rapidly enter the mainstream based on its improved specifications. For instance, the USB has become the most popular peripheral interface standard for consumer products during the past few years. The earlier IEEE 1284 parallel interface is no longer able to be supported in most devices. Thus, it is becomes a possibility that long lifetime systems will suffer the problem of interface mismatch because of these modern peripherals.
- *Communication bus obsolete.* In this case, the communication bus is considered as an on-board or on-chip bus, which is the link between each component standing in the system. It is not possible to support the previous Industry Standard Architecture (ISA) bus which is replaced by the Peripheral Component Interconnect (PCI) bus. For a hardware component, backward compatibility is not always guaranteed unless it incorporates an extra hardware bridge between two buses. The System-on-chip (SOC) design has also suffered from the obsolescence associated with the communication bus. For example, the On-chip Peripheral Bus (OPB) has been replaced by the Processor Local Bus (PLB) [8] for the Xilinx FPGA on-chip bus.
- *Component obsolete.* A component is a product provided by the vendor, which for the majority, contains some unique properties and cannot be replaced by a product from other vendors. Component obsolescence is a severe case since it frequently occurs. Industry experts estimated that over 200 000 components from over 100 manufacturers became obsolete in 2000 [9].
- *Others.* Obsolescence issues such as obsolete development tools and test systems etc. must all be faced by designers.

B. Designer

Lower prices or better circuit technology could offer the opportunity for designers to replace the legacy components or even an entire system, so that they can reduce cost and increase value [3]. However, it has not proved to be easy to enable either migration or replacement to occur within the different technologies as this will involve costly hardware and software redesigns.

C. Customer

It is a self evident truth that the customer always wants more. Manufacturers must reengineer their products to provide new features required by customers, to incorporate newer technologies and standards. Exciting new technologies can result in a better form and fit for a specification. Functions will require to be changed as will the bugs contained within the system and these require that the legacy system be re-engineered [3]. This is always costly and time consuming.

IV. Proposed Solution

We propose the following solutions in order to mitigate the trouble for embedded system maintenance issues discussed in section III. We intend to proof its positive impact on maintenance for results presented in section VII.

A. FPGA platform

FPGA is an integrated circuit designed to be configured by the customer or designer after manufacturing. The FPGA configuration is generally specified using a hardware description language (HDL). HDL is a high level system description, allowing it to be translated and implemented by the synthesis tools onto the hardware.

FPGA is an ideal system platform for its flexibility and reconfigurability. The hardware can be modified on-chip according to the different requirements of the customers. The process of reconfiguration is so convenient that an engineer could even upgrade and debug the system remotely. The cost for system maintenance is thus significantly reduced.

B. Device and vendor independent soft IP

For FPGA system, designer should care about system migration (portability) issue, both for hardware and software. While FPGA is only a hardware platform, the system hardware actually consists of a set of components, called IPs. They can be in soft or firm (technology-independent net lists) form. A soft IP is a hardware specification at the register transfer level (RTL), which is a set of synthesizable code written in HDL. The software is taken as a reference in this case. As the software has no risk of obsolescence and can be reconfigured. In addition, some offer very good portability between hardware platforms. A soft IP has some similar characteristics to those associated with software and is thus a rather suitable form for an FPGA system component, since HDL can be written in a technology-independent manner and synthesized into gate level. The synthesis tools will handle the configuration for different technologies. In order to ease the process of IP migrating to new technologies, a synthesizable device and vendor independent soft IP are to be preferred. Its advantages include flexibility, portability and reusability [10]. For example, if one soft microprocessor IP has been originally implemented on a Xilinx VirtexII FPGA chip then the IPs and software

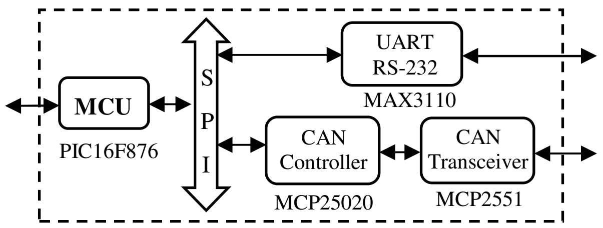

Figure 1. Block diagram of COTS based CAN controller system

applications can be migrated to an Altera Cyclone III chip without causing any difficulties.

V. EXPERIMENTAL SYSTEM AND DESIGN CASES

This section describes this particular experimental system and the four cases which were to be analyzed.

CAN is an industrial bus standard designed to allow microcontrollers and devices to communicate with each other. This project started from a CAN controller system used in industrial construction machinery. In this case, cost, performance and power consumption are not critical issues. However, this type of long lifetime system requires greater consideration to be given to the maintenance.

The microprocessor (microcontroller) and the CAN controller are two key components for this system. At this point two major design platform methods are mentioned, namely the COTS IC platform and the FPGA platform. FPGA system mentioned in this paper is an IP based design system. There is a wide range of choices for the COTS microcontrollers and CAN controllers IC from different vendors within the marketplace.

However, for the FPGA platform, the soft microprocessors are divided into two categories:

- *Vendor dependent soft microprocessors:* Such kinds of soft microprocessors are usually provided by the FPGA vendors, so it is not possible to implement them on any other vendor's devices. E.g. Xilinx MicroBlaze and Altera Nios II.
- *Vendor independent soft microprocessors:* Unlike the vendor dependent soft microprocessors, these have no restrictions and can be implemented on any vendors' devices. E.g. ARM Cortex-M1 and OpenRISC from OpenCores.

A. Case 1: COTS IC based CAN controller system

For the previous CAN controller system, a Microcontroller (MCU) was implemented onto board. The UART controller and the physical interface circuit RS-232 were integrated as one chip for the peripheral interface. A CAN controller and its physical interface circuit CAN transceiver were implemented on board for the CAN bus protocol. The system board architecture with its relevant components can be seen in Fig. 1.

B. Case 2: vendor specific FPGA system

A vendor specific design case based on the Xilinx Spartan3E FPGA is shown in Fig. 2. The MicroBlaze soft processor controls the UART lite and XPS-CAN via the Processor Local Bus (PLB). The MicroBlaze Debug Module (MDM) is used for system debugging. The

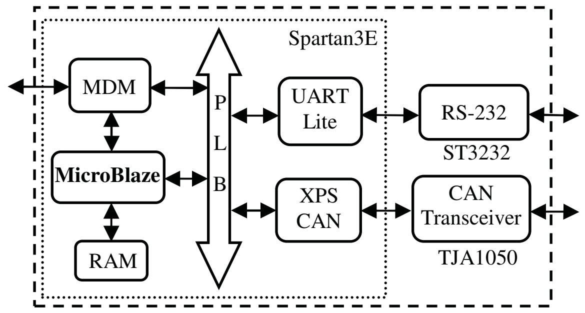

Figure 2. System architecture of Xilinx specific FPGA system

development tools, FPGA devices and IPs are all provided by Xilinx. It is easy to perform system development since the majority of IPs are verified and can be plug-and-play. The whole design and verification process can be executed with a Xilinx tool set. The RS-232 and CAN transceiver physical interface circuits are also integrated on board.

C. Case 3: vendor and device independent FPGA system

A vendor and device independent system is a "soft" system which can be implemented on any FPGA device. The IP could be open-source licensed or provided by third party IP providers. Such a case is based on OpenCores soft IPs. OpenRISC 1200 (OR1200) is a general purpose open source RISC CPU, which has already been verified as running on many vendors' devices and can be free downloaded and modified by any individual.

The CAN controller system based on the OR1200 is shown in Fig. 3. The board structure is the same as in case 2, while the FPGA on-chip architecture is different. Every IP in the system is open source licensed in addition to being vendor and device independent. They communicate with each other via a wishbone bus. A debugger is used for debugging and software downloading. Different JTAG cables can be used on different vendor devices and the entire system can be synthesized by using any synthesis tool. The GNU toolchain [11] which is running on a PC including a compiler, simulator, debugger etc, is used to support C software development as well as system debugging.

D. Case 4: mixed FPGA system

This solution is comprised of a mixture of vendors' specifics and an OpenCores platform. The board structure is the same as in case 2 and is shown in Fig. 4. A bridge IP is incorporated between the Xilinx PLB and the

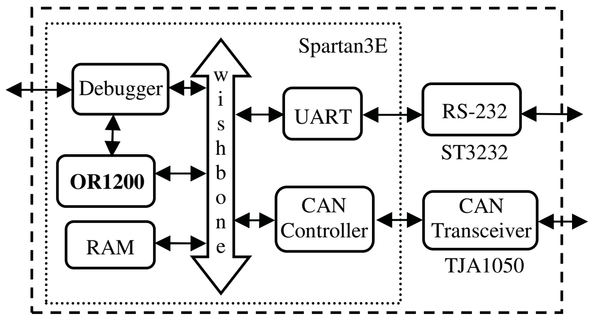

Figure 3. System architecture of OpenCores based FPGA system

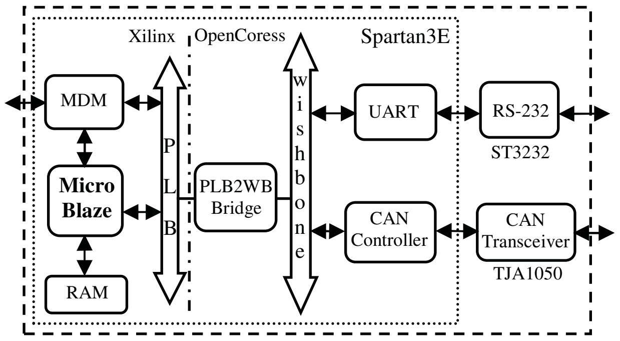

Figure 4. system architecture of Xilinx and OpenCores mixed FPGA system

OpenCores wishbone bus. The open-source licensed soft IPs such as the CAN controller and the UART can then be integrated into the system as a peripheral core for the Xilinx system. All of the design and verification processes can be conducted in a Xilinx development environment.

VI. Case study

We take a risk analysis for all cases described in section V. A number of potential risk scenarios are identified. No probability is attached to the occurrence of each risk, but the consequences for the maintenance work are classified and evaluated, as shown in Fig. 5.

A. Risk scenarios

For the different platform cases presented in section V, the system maintainability is evaluated by analyzing several potential risk scenarios. These scenarios have been developed according to the general problem issues discussed in section III.

- *Microprocessor obsolescence:* Microprocessor is the heart of an embedded system. If it becomes obsolete, then there could be serious consequences.
- *Peripheral interface obsolescence:* A peripheral interface standard can be obsolete. The RS-232 interface in the system has the risk of obsolescence, including a UART controller chip, the physical interface circuit and connector.
- *Communication bus obsolescence:* The communication bus has a connection with every component in the system. Its obsolescence will soon lead to the obsolescence of all associated components.

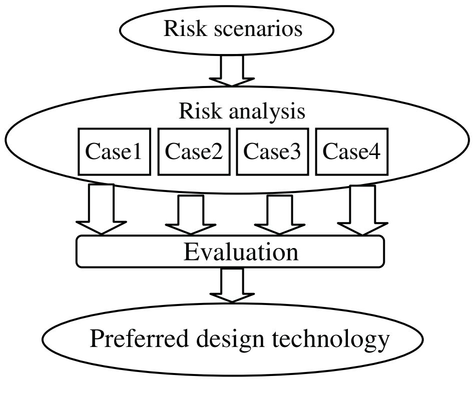

Figure 5. Risk analysis for different cases

- *Better circuit technology migration*: Better performance, lower price or being friendlier towards the environment would force the system to migrate to a new circuit technology.
- *FPGA vendor device migration:* Vendor portability is a special issue for the FPGA system. For example, if the FPGA vendor stops providing the devices (obsolescence), in this situation, the system is forced to migrate from one vendor device to another.
- *Function change requirement:* For different requirements, add, delete or modification of functions are inevitable. For example, some systems might require an Ethernet interface for data transportation.

B. Consequences

Some of the consequences with regards to the risk scenarios are classified.

- *Major board redesign:* This work is to redesign the board, including replacing or adding components, modifying the on-board bus system etc., which is costly and time consuming and can almost be equivalent to designing a new board.
- *Minor board redesign:* The redesign of a minor part of the board, including replacing or adding physical interface circuits, redefining the pins for chips, changing the connector etc., which require much lower design efforts as compared to a major board redesign.
- *Driver redesign:* This is the redesigning of the software drivers' work in order to be consistent with hardware changes.
- *Interface modification:* Modify the soft IPs' communication bus interface protocol or add an interface converter.
- *Vendor restriction:* Vendor restriction is specified for an FPGA platform. It means the FPGA devices can be changed, but this is restricted to the same vendor.
- *Vendor independent:* In contrast with vendor restriction, it does not have any restriction regarding the vendors. Any vendor device could be used.
- *Major system redesign:* It means that there is a whole redesign of the FPGA system, including the on-chip hardware and software driver redesign.
- *Minor system redesign:* Redesign parts of an FPGA system, including parts of the on-chip hardware and software driver redesign.

VII. Result

Table I presents the results of the risk analysis for design cases defined in section V. Following is the explanation of table I for each of the risk scenarios:

- *Microprocessor obsolescence:* For a COTS product based platform, if the MCU becomes obsolete, the entire system will become obsolete. It results in a major board redesign and driver redesign for a new MCU system. While the FPGA microprocessor is

TABLE I. CONSEQUENCES OF DIFFERENT RISK SCENARIOS

Risks	Consequences	Case			
		1[a]	2[b]	3[c]	4[d]
Microprocessor obsolescence	Major board redesign	√	-	-	-
	Driver redesign	√	-	-	-
Peripheral interface obsolescence	Major board redesign	√	-	-	-
	Minor board redesign	-	√	√	√
	Driver redesign	√	√	√	√
Communication bus obsolescence	Major board redesign	√	-	-	-
	Interface modification	-	√	√	√
Better circuit technology migration	Major board redesign	√	-	-	-
	Minor board redesign	-	√	√	√
	Driver redesign	√	-	-	-
	Vendor restriction	N/A	√	-	√
	Vendor independent	N/A	-	√	-
FPGA vendor device migration	Major system redesign	N/A	√	-	-
	Minor system redesign	N/A	-	-	√
	Minor board redesign	N/A	√	√	√
Function change requirement	Major board redesign	√	-	-	-
	Minor board redesign	-	√	√	√
	Driver redesign	√	√	√	√

a. COTS IC based CAN controller system

b. Vendor specific FPGA system

c. Vendor and device independent FPGA system

d. Mixed FPGA system

described as a synthesizable soft code, such a special form completely eliminates the risk of microprocessor obsolescence. The legacy IP can still be implemented on a recent FPGA device, which can solve the component obsolete issue described in section III.A.

- Peripheral interface obsolescence: The RS-232 serial interface in the system has the risk of obsolescence. If the MCU on the COTS platform does not support the new peripheral interface standard, it should be replaced or a new interface controller should be integrated together with a physical interface circuit. Both situations will lead to major board redesign. While for the FPGA platform, the interface controller can be changed by replacing a soft controller IP on-chip. The physical interface circuit (voltage converter etc.) and connector must also be changed which is a significantly easier task. Software driver redesign is necessary in order to adapt the new interface controller to the system. This will ease the peripheral interface obsolete issue of section III.A.

- *Communication bus obsolescence:* For the COTS platform, the whole system has to be redesigned since every component which is associated with the bus is obsolete. The FPGA soft IP does not have any risk of obsolescence, but new IPs will face interface mismatch problems with an old communication bus. Interface modification is required if the new IPs are integrated into the system, which will ease the communication bus obsolescence describe in section III.A.

- *Better circuit technology migration:* It is difficult for the COTS IC platform to benefit from better circuit technology requiring a major board and driver redesign. However, all the "soft" systems on the FPGA proposed in section V have the capability of accommodating to new technology, only requiring minor board redesigns to redefine the pins for the new FPGA chip. However, a technology migration with regards to cases 2 and 4 is restricted to using the same vendor's device. Case 3 is the best choice regarding to maintenance issue described in section III.B because of its device independency.

- *FPGA Vendor device migration:* Vendor portability is an issue which is only relevant for FPGA systems. As has been mentioned previously, case 2 is a vendor specific solution and thus it is not possible to migrate it to other vendors' devices. If the vendor ceases to provide the devices, e.g. vendor bankrupt, the only choice is to redesign a whole new system for another vendor's device. There is a better situation associated with case 4 as parts of the system are vendor independent and with the assistance of a new bus bridge, these parts could be migrated to other vendors' device. The design effort compared to case 2 is significantly lower. For example, the Avalon to wishbone bus bridge IP could be used if the system is migrated to the Altera's device. Case 3 is a totally vendor independent system and thus offers the best portability from the FPGA platform cases and it would be easy to implement the vendor device migration.

- *Function change:* The function must be changed for any new function requirement or for removing bugs in the current component or system. It is a costly task for a COTS platform, because it results in a major board redesign. Due to FPGA's reconfigurabilty, the function in the form of a soft IP can be added, deleted or modified on-chip. In some situations, a physical circuit is required, such as an Ethernet controller. Some function changes do not even require a board redesign but only a modification on-chip, if the FPGA is sufficiently large to accommodate the function, such as a video decoder. In this case, it will ease the maintenance issue described in section III.C.

VIII. DISCUSSION

According to the evaluation in section VII, Fig. 6 can be used to encapsulate our conclusions.

COTS IC solution has no reconfigurability and portability. The MCU and CAN controller are in the fixed hard form provided by the IC providers. The platform defines its specific drivers and software applications. After the completion and release of the system design, it becomes difficult to make any further changes. Therefore, it always requires a major board and driver redesign as shown in table I, the implication of which is that this will involve high maintenance costs. However, the benefits of the COTS IC are its mature technology and market. COTS

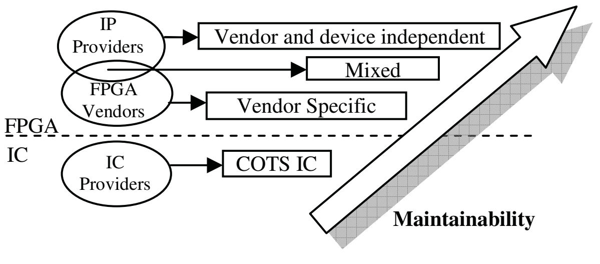

Figure 6. System maintainability model for different design technologies

IC design technology is now the dominant design method for an embedded system. Compared to the FPGA platform, it is more attractive with respect to higher performance, lower power consumption and cost.

The FPGA system is described using a high level language and is implemented on a single chip. If sufficient space is reserved on the chip, a system function could easily be changed using EDA tools. Sometimes the function requires small changes of the physical circuit and connectors as only a minor board redesign as shown in table I. Thus the embedded system built on the FPGA platform is very promising when maintenance issues are taken into consideration.

Vendor specific design technology is possibly most widely used on an FPGA platform. Provision of the IPs for the system is by the FPGA vendors. IPs such as this are of reliable quality and it is very easy to integrate them into the system. Designers can obtain adequate support and guarantees. The system might be portable within the same vendor's devices. However, such system design technology is unable to eliminate the risk if the vendor ceases to provide the devices (obsolescence). The reason for mixed solution has better portability is associated with the device independent parts. A designer can make an effort to integrate the OpenCores parts to other vendors' system by using a different bus bridge. Such mixed platform benefit from vendor specific as well as vendor and device independent IPs, which can be a compromised alternative for an embedded system design.

Although software development and related tools are not included in the risk analysis, they also represent an unavoidable part in today embedded systems development. Software application of embedded system usually programmed in high level language (e.g. C), which always contain some processor dependent code, such as driver, interrupt control etc. If the microprocessor is replace by a different one, the software has to be modified consistently. The development environment such as compiler and library can also be different. Such software modification and environment change will result in system re-verification, which is time consuming.

The FPGA platform implemented with vendor and device independent soft IPs is a preferred solution from the viewpoint of system maintenance, since it can solve all the maintenance problem issue described in section III. The whole system including the hardware and software can be accommodated into any vendor device with any technology. The software application and GNU toolchain can be used on new system device when doing the system migration. Thus this eliminates the risk of device obsolescence from providers and it is also possible to benefit from the use of new circuit technology or lower cost hardware. Functions can also be modified for customer's requirement. Therefore, such method could compared to the traditional way for a COTS IC based embedded system design technology, the system on an FPGA in combination with soft IPs has much higher maintainability according to the results shown in table I.

It is well known that FPGA's circuit technology and performance are not always growing as fast as a COTS IC component, but this could be tolerated by control, monitoring or communication systems, which are manufactured in low volumes [12]. Many of these systems do not require the newest state of the art technology and would be in use, preferably, for more than 10 years, such as the case for the CAN controller system.

The soft IP market is, at present, not sufficiently mature and for this OpenCores based platform, the quality of the open source IP is not satisfactory, since the designer can hardly get guarantee from providers. Due to lack of intensive verification, the IPs always contain bugs from our experience. However, it is to be expected that in the future, the high reliability vendor and device independent soft IP could be delivered by third party IP providers. Such IPs will become very good resources for designing a long lifetime embedded system.

IX. Conclusion

This paper has discussed the various potential risks and their consequences during product maintenance work. This is especially relevant for an embedded system which has long lifetime expectancy. It was proposed that an FPGA platform be used to replace the COTS IC platform for the system design. Different CAN controller system cases have been evaluated using a number of risk scenarios. The results show that an FPGA platform with vendor and device independent soft IP is the preferred design technology. The result of our research could prove to be useful for designers who might be face difficulties in relation to maintaining a long lifetime embedded system.

References

[1] N. Audsley, I. Bate, A. Grigg, "Portable code: reducing the cost of obsolescence in embedded systems," Computing & Control Engineering Journal, 1999, pp. 98–104.

[2] P. Sandborn, "Trapped on Technology's Edge," IEEE Spectrum, 2008, pp. 42–58.

[3] V. K. Madisetti, "Reegineering digital systems," IEEE Design & Test of Computers, 1999, pp. 15–16.

[4] E. F. Hitt, J. Schmidt, "Technology obsolescence (TO) impact on future costs," Proceedings of 17th digital Avionics Systems Conference, 1998, pp: A33 – 1-7 vol.1.

[5] A. Meyer, L. Pretorius, J.H.C. Pretorius, "A model to manage electronic component obsolescence for complex or long life systems," Proceedings IEEE International Engineering Management Conference, 2004, pp. 1303–1309.

[6] P. A. Sandborn, F. Mauro, R. Knox, "A Data Mining Based Approach to Electronic Part Obsolescence Forecasting," IEEE Transactions on Components and Packaging Technologies, 2007, pp. 397–401.

[7] R. Velazco, L. Anghel, S. Saleh, "A methodology for test replacement solutions of obsolete processors," 9th IEEE On-Line Testing Symposium, 2003, pp. 209–213

[8] PLB v3.4 and OPB to PLB v4.6 System and Core Migration User Guide.http://www.xilinx.com/support/documentation/sw_manuals/edk10_mg_ug.pdf

[9] "ElectronicsTalk," Aug. 19, 2003. http://www.electronicstalk.com/news/tex/tex489.html, Texas Instruments. Obsolescence Policy Gains Period of Grace.

[10] R. Saleh et al., "System-on-Chip: Reuse and Integration," Proceedings of the IEEE, 2006, pp. 1050–1069

[11] GNU toolchain, http://www.opencores.org/openrisc,gnu_toolchain

[12] J. Torresen, T.A. Lovland., "Parts Obsolescence Challenges for the Electronics Industry," IEEE Design and Diagnostics of Electronic Circuits and Systems, 2007. pp 1.