Innovazione nello sviluppo di piattaforme informatiche applicabili ai sistemi per la sicurezza

Innovation in the software development platforms for safety critical systems

Dott. Ing. Roberto NAPPI[*], *Dott. Ing. Giovanni* FLORIO[**]

1. Introduzione

I risultati delle attività di ricerca e sperimentazione presentati in questo articolo si collocano nell'ambito della definizione di strumenti e metodologie di processo innovativi per la programmazione dei dispositivi elettronici digitali di controllo (sistemi *embedded*), basati su FPGA[1] per applicazioni *safety* e *mission critical*. L'oggetto di tali risultati è costituito dalla Piattaforma FAD&T (*FPGA Automatic Development & Testing Platform*) impiegabile come strumento per la progettazione e lo sviluppo automatizzato del *firmware* dei sistemi embedded, riducendone i costi di produzione, a parità di grado di fidatezza atteso.

2. Definizione del contesto delle attività di ricerca e sperimentazione

La presenza di dispositivi elettronici digitali di controllo in ambiti industriali diversi è in costante crescita. Con il termine "sistema embedded" si identificano tutti quei sistemi elettronici di elaborazione progettati per una specifica applicazione. Tali sistemi sono detti *embedded* in quanto risultano inseriti nello stesso ambiente che "monitorano", mediante sensori, e che "controllano", mediante attuatori. Tali attività di monitoraggio e controllo sono rese possibili grazie al codice da essi usato (*firmware*), impiegato per eseguire elaborazioni basate sui dati provenienti dai sensori e applicare attuazioni sulla base delle decisioni prese, secondo quanto mostrato in fig. 1.

La presenza dei sistemi *embedded,* in diversi settori industriali, è rilevante, così come mostrato dal grafico di fig. 2, da cui si riscontra che circa la metà del costo finale di un prodotto elettronico è rappresentato dai sistemi *embedded* in esso usati.

[*] Systems Engineering T&T srl.

[**] Marketing T&T srl.

[1] L'FPGA (Field Programmable Gate Array) è un dispositivo elettronico integrato che contiene porte logiche interconnesse per elaborare dati e implementare funzioni. Le interconnessioni hardware sono programmate "on field" dall'end-user dell'FPGA piuttosto che "in foundry" dal produttore.

1. Introduction

The results of both research and experimentation activities presented in this paper are related to the definition of innovative tools and methodologies to program embedded systems based on FPGA[1] in safety and mission critical applications. The main result of these activities is the FAD&T (FPGA Automatic Development & Testing) Platform applicable as tool for automated design and development of firmware of the embedded systems, providing a cost reduction with the same expected reliability.

2. Context definition of the research and experimentation activities

The use of electronic control devices is constantly growing in several industrial areas. The term "embedded systems" identifies all the elaboration electronic devices, designed for a specific application. These systems are called *embedded* as they are inserted in the same environment that they "monitor" by means of sensors and that they control by means of actuators. These monitoring and controlling activities are possible thanks to the code they use (firmware), used to elaborate data coming from sensors and perform implementations based on the decisions taken, as shown in fig. 1.

The embedded systems' presence in different industrial areas is significant as shown in the graph of fig. 2 from which one can see that approximately 50% of the whole electronic device cost is related to the embedded systems used in them.

A "system" is generally able to deliver a certain number of services through the implementation of appropriate functionalities. In particular, "critical systems" are

[*] Systems Engineering T&T Srl.

[**] Marketing T&T Srl.

[1] FPGA (Field Programmable Gate Array) is an integrated electronic device that contains hardware programmable interconnections between theirs logic gates to elaborate data and implement functions. The hardware interconnections are programmed "on field" by the end-user of the FPGA instead of the producer of the FPGA "in foundry".

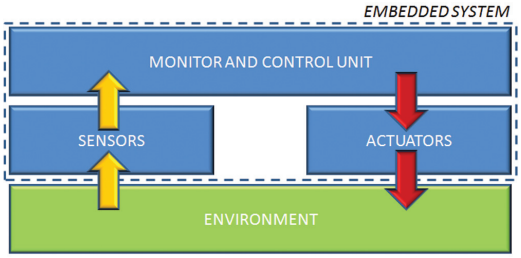

Fig. 1 - Attività di monitoraggio e controllo espletata dai sistemi embedded rispetto all'ambiente in cui sono "immersi". *Monitor & control activities of the embedded systems.*

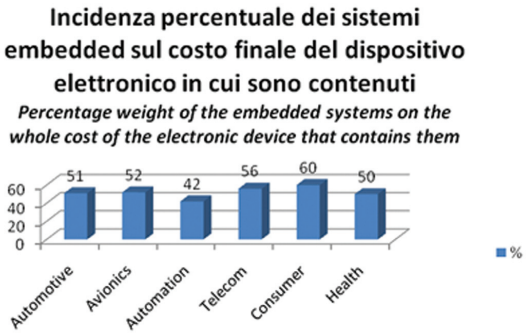

Fig. 2 - Peso percentuale dei costi dei sistemi embedded rispetto a quelli di produzione dell'intero sistema elettronico nel quale sono inseriti. *Percentage weight of the embedded systems on the whole cost of the electronic device that contains them.*

In generale, un sistema è in grado di fornire un determinato numero di servizi mediante l'implementazione di opportune funzionalità. In particolare, i "sistemi critici" sono tali quando, non fornendo servizi secondo le modalità attese, possono causare danni a cose oppure a persone. I sistemi *safety critical*, i cui fallimenti possono provocare incidenti, perdite di vite umane o seri danni ambientali[2], ed i sistemi *mission critical*, i cui malfunzionamenti possono causare il fallimento di alcune attività con obiettivi diretti[3], sono entrambi particolari tipologie di sistemi critici. La proprietà più importante per un sistema critico è la fidatezza (*dependability*) definita come la proprietà del sistema che permette di porre fiducia, in modo dimostrabile, nei servizi che esso fornisce. Il termine fidatezza è usato per indicare complessivamente le proprietà di affidabilità, disponibilità, manutenibilità e sicurezza definite in fig. 3, che sono intrinsecamente legate tra loro.

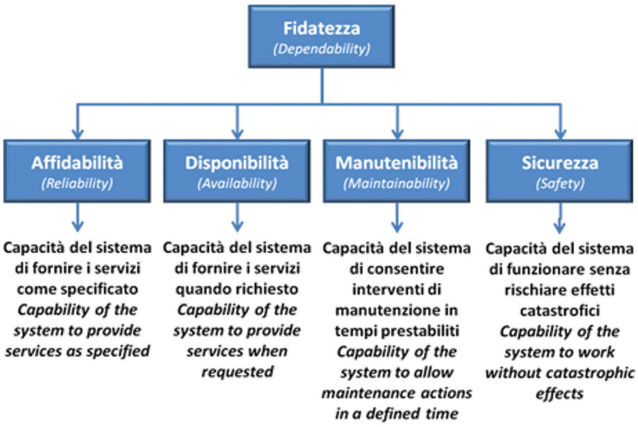

Fig. 3 - Schema delle caratteristiche di un sistema critico. *Properties scheme of a critical system.*

such when they produce damages to things or persons, not providing services according to the expected modalities. Safety-critical systems, whose failures can cause accidents, loss of human lives or serious environmental[2] damages and mission-critical systems, whose malfunctions may cause failures in activities with specific purposes[3] are both specific kinds of critical systems. The main property of a critical system is the reliability defined as the system property that allows a demonstrable level of trust in the services that it provides. The term "dependability" is used to indicate the overall reliability, availability, maintainability and, safety properties defined in fig. 3, that are strictly related to each other.

The designers of critical systems must search the right trade-off between system performances and dependability. High levels of reliability can generally be achieved to the detriment of performances however. Due to the high costs for design, development and validation processes, the increase in reliability of a system can significantly increase development costs. Often, validation costs are higher than design and development costs. Moreover, a lot of costs sustained by producers are related also to assessment activities, typically performed by independent agencies, for the evaluation of the dependability level obtained for the critical system before its release.

Fig. 4 shows the qualitative relationship between system dependability and sustained costs to obtain it. The more the system needs to be reliable the more it is "expensive", both

[2] Esempi di sistemi *safety critical* sono: i sistemi di controllo dei velivoli, i sistemi di controllo dei processi nelle fabbriche chimiche o farmaceutiche, e i sistemi di instradamento del traffico ferroviario.

[3] Un esempio di sistema *mission critical* è il sistema di navigazione di un veicolo spaziale.

[2] Examples of safety critical systems are: airplane control systems, control systems of production processes in chemical farms or in pharmaceutical farms, railway traffic routing systems.

[3] An example of a mission critical system is the navigation system of a spacecraft.

I progettisti dei sistemi critici sono tenuti a trovare un equilibrio tra prestazioni del sistema e fidatezza: generalmente è possibile raggiungere alti livelli di fidatezza a scapito però delle prestazioni. A causa dei costi di progettazione, implementazione e validazione, l'aumento di fidatezza di un sistema può far crescere significativamente il costo di sviluppo. Molto spesso i costi di validazione sono più elevati dei costi di progettazione e sviluppo. In più, una buona parte dei costi sostenuti dai produttori sono relativi alle attività di valutazione svolte da agenzie indipendenti per la determinazione del livello di fidatezza del sistema critico, prima del suo rilascio.

La fig. 4 mostra la relazione qualitativa tra fidatezza e costi sostenuti per ottenerla. Più fidato si vuole un sistema e più occorre "spendere", sia per implementare le misure di progetto, in linea con tali obiettivi, sia per dimostrare il livello di fidatezza raggiunto. A causa della natura esponenziale della curva costo/fidatezza, si rileva che un sistema fidato al 100%, in linea di principio, comporta costi infiniti.

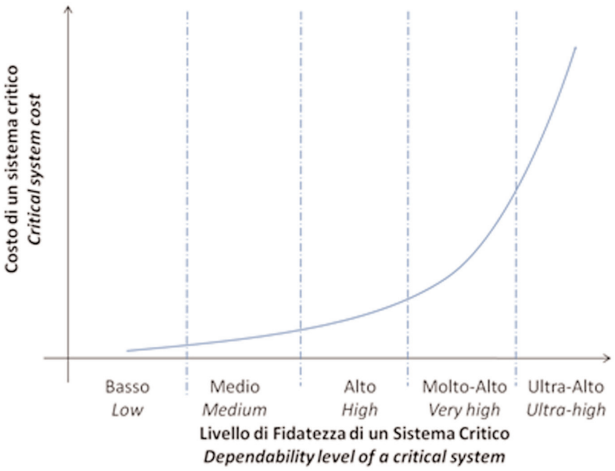

Fig. 4 - Curva Costo/Fidatezza. *Cost/Dependability curve.*

Pertanto, per ridurre le conseguenze del fallimento di un servizio e gli elevati costi di produzione dei sistemi critici è necessario utilizzare processi specifici e strumenti complessi per il loro sviluppo. È in tale contesto che si collocano i risultati delle attività di ricerca e sperimentazione illustrate nel presente articolo.

Le soluzioni progettuali utilizzate nel tempo hanno previsto l'utilizzo di microprocessori sempre più potenti, più performanti e dotati di *firmware* di elaborazione particolarmente complessi, al fine di gestire gli elevati livelli di fidatezza richiesti. Ciò ha prodotto un incremento dei costi e una elevata "castomizzazione" del prodotto sviluppato, a causa anche della scarsa riutilizzabilità di tali dispositivi in contesti diversi da quelli iniziali. Per risolvere tale problema, la tendenza attuale nello sviluppo di sistemi elettronici "critici" è quella di associare al microprocessore un ridotto numero di funzionalità, delegando ad altri componenti elettronici, quali le FPGA (*Field Programmable Gate Array)*, le rimanenti funzionalità, dando luogo al passaggio dalla gestione "accentrata" (fig. 5a) del controllo, ad una sua gestione "distribuita" (fig. 5b).

Con l'approccio distribuito la determinazione del livello di fidatezza è semplificata, grazie alla riduzione della complessità così ottenuta, mentre la fidatezza complessivamente raggiunta risulta migliorata rispetto all'approccio "accentrato", dal momento che un malfunzionamento "periferico", essendo circoscritto, non impedisce alla restante struttura del sistema di continuare ad erogare correttamente i servizi previsti. In particolare, l'impiego di dispositivi FPGA si rende utile per la maggiore velocità elaborativa rispetto ai microcontrollori, quando sono richieste elaborazioni in tempo reale, e per la maggiore fi-

to implement the project measures, in line with these objectives, and to prove the reliability level obtained. Due to the exponential nature of the cost/dependability ratio, a 100% reliable system in principle has infinite production costs.

Therefore, in order to reduce the consequences of a wrong service and high production costs of critical systems specific processes and complex tools should be used. The results of the research and experimentation activities presented in this paper can be set in this context.

In the past, the design solutions have provided for the use of increasingly powerful and performing microprocessors with particularly complex processing *firmware* in order to manage the high level of requested dependability. This produced a cost increase with an high "customization" of the product developed due to low reusability of such devices in different contexts compared to the initial ones. To solve this problem, the trend in the development of "critical" electronic devices today is to associate the microprocessor with a reduced number of functionalities, leaving other electronic components such as FPGA (*Field Programmable Gate Array*) to deal with the remaining functionalities, therefore changing from a "centralized" management control to a "distributed" management (fig. 5b).

With a decentralized approach, the determination of the dependability level of a system is simplified thanks also to the reduction of the complexity achieved, whereas the overall dependability is improved compared to the "centralized" approach, given that a "peripheral" localized malfunction, does not in fact prevent the system's remaining structure to continue delivering the expected services correctly. In particular, the use of FPGAs is useful due to the greater elaboration speed, compared with microcontrollers, when real-time elaborations are necessary and for the greater depend-

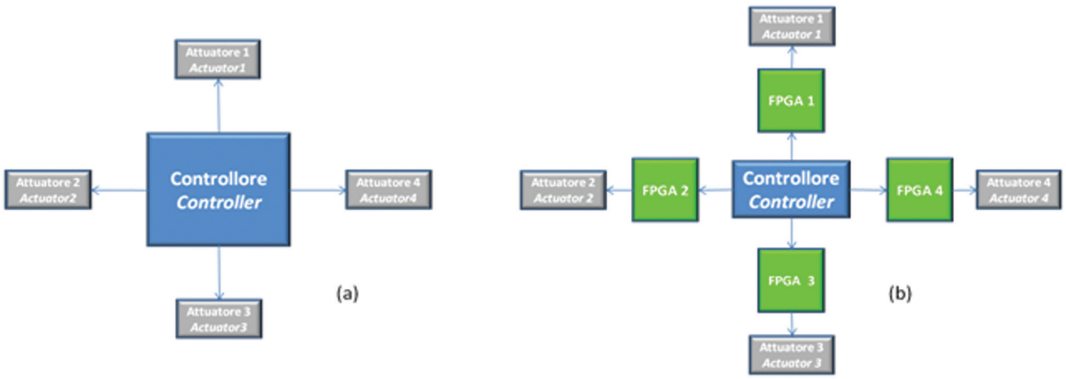

Fig. 5 - Gestione "accentrata" (a) e "distribuita" (b) del controllo. *Centralized (a) and decentralized (b) control management.*

datezza della programmazione hardware[4] usata per le FPGA rispetto alla programmazione strutturata usata per i microcontrollori.

3. Stato dell'arte e confronto con FAD&T (FPGA Automatic Development & Testing) Platform

Nell'ambito della progettazione e della realizzazione del firmware dei sistemi embedded è possibile raggruppare le attività da svolgere secondo tre livelli principali:

- System Modeling Level;
- Firmware Design and Development Level;
- System Integration Level.

Attualmente esistono diversi strumenti di ausilio allo sviluppo del firmware, non necessariamente certificabili per fidatezza, che agiscono separatamente nei diversi tre precedenti livelli. A livello *System Modeling* esistono tools che partendo dai requisiti espressi in maniera formalizzata sono in grado di fornire un modello del sistema da cui poter verificare se i requisiti disponibili sono in grado di definire tutti gli obiettivi per cui il sistema deve essere realizzato. A livello *Firmware Design and Development* è possibile produrre in modo automatizzato il codice, generalmente in un solo linguaggio di programmazione, che implementa i requisiti del sistema, forniti in maniera formalizzata. A livello *System Integration*, generalmente su singola piattaforma hardware, è possibile generare protocolli di test con esecuzione automatizzata a partire dal modello formalizzato del sistema.

La Piattaforma FAD&T, presentata in questo articolo consente, invece, di integrare in un unico strumento tutti e tre i precedenti livelli dello sviluppo del firmware. Infat-

[4] L'Hardware Description Language (HDL) è uno strumento per progettare dispositivi elettronici integrati. È usato per definire le funzionalità del dispositivo e gli aspetti architetturali hardware che le implementano.

ability of hardware programming[4] used for the FPGAs compared to the structured programming used for microcontrollers.

3. State of the art and comparison with FAD&T (FPGA Automatic Development & Testing) Platform

It is possible to group all the activities needed for the design and development of firmware of the embedded systems in three fundamental levels:

- System Modelling Level;
- Firmware Design and Development Level;
- System Integration Level.

Nowadays, there are several tools useful to develop firmware, not always certifiable for dependability, and they work separately in each of the three previous levels. At *System Modelling* Level there are tools that, starting from formalized requirements, give a system model to verify that all the defined requirements satisfy the system purposes. At *Firmware Design and Development* Level it is possible to generate the code automatically, generally using a single hardware programming language that implements the system requirements. At *System Integration* Level, generally on a single hardware platform, it is possible to generate test protocols automatically and test executions starting from the formalized system model.

Instead, the FAD&T Platform presented in this paper allows the integration of the previous three firmware development levels in a standalone tool. In fact, at first step

[4] The hardware description language designs integrated electronic devices. It is used to define the system functionalities and the architectural solutions that implement them.

ti, è in grado di produrre automaticamente il modello di sistema, a partire dai suoi requisiti, e di testarlo sempre automaticamente. Successivamente FAD&T è in grado di generare e testare automaticamente il firmware, producibile in linguaggi di programmazione anche diversi, che implementa il modello di sistema precedentemente ottenuto. Infine, grazie alla sua interfaccia hardware, multipiattaforma e configurabile, consente di testare, in maniera automatizzata, l'integrazione tra il firmware prodotto in precedenza e l'hardware FPGA, sempre sulla base del modello generato e testato in precedenza. L'impiego di un approccio integrato, verticale e parallelizzato, per i tre livelli dello sviluppo, secondo quanto indicato in fig. 6b, rispetto all'attuale stato dell'arte dove questi livelli sono invece processati separatamente, orizzontalmente e sequenzialmente, secondo quanto indicato in fig. 6a, consente di orientare alla testabilità l'intero processo di sviluppo di un sistema embedded (*Design for Testability*), abbattendo i tempi di produzione grazie alla parallelizzazione delle attività di sviluppo e testing, e aumentando la

it automatically generates the system model starting from its requirements and automatically tests the system model. In the next step, FAD&T Platform automatically generates the firmware, in several different hardware programming languages, and implements the system model previously generated and tested. Lastly, FAD&T allows to automatically test the firmware that has been integrated in the FPGA, thanks to its multiplatform and configurable hardware interface and using the previously generated system model. This integrated process, based on a parallel and vertical approach for the three levels of firmware development, as indicated in fig. 6b, compared with the state of the art where these levels are instead separately processed in sequential and horizontal manner, as indicated in fig. 6a, allows to orient for testability the whole development process of the embedded system (*Design for Testability*), providing both costs and time reduction, thanks to paralleling development and testing activities, increasing the dependability of the product also due to the three levels of testing applied.

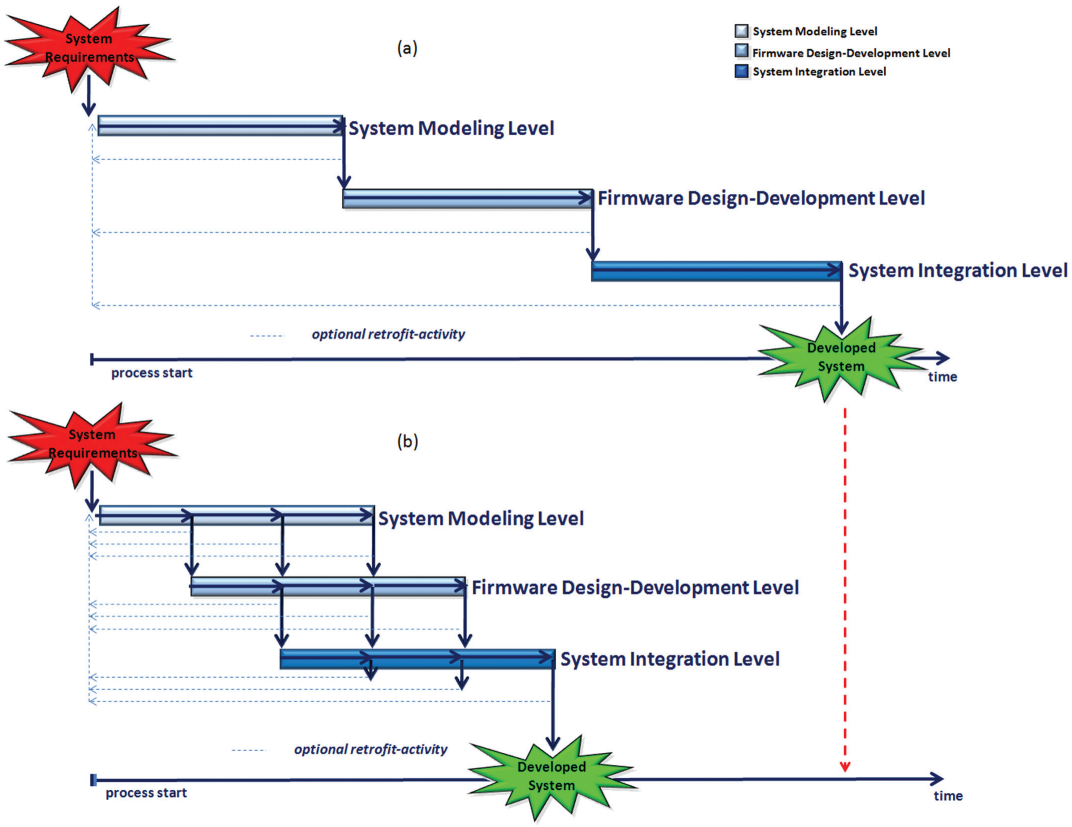

Fig. 6 - (a) Processo di Sviluppo del firmware di tipo *orizzontale-sequenziale* (Approccio tradizionale) - (b) Processo di sviluppo del firmware di tipo *verticale-parallelizzato* (Approccio innovativo). *Vertical-parallelized firmware development process (Innovative approach) - (b) Vertical-parallelized firmware development process (Innovative approach).*

fidatezza del prodotto grazie anche all'applicazione dei tre livelli di testing indicati.

L'attività di verticalizzazione e parallelizzazione dello sviluppo è ottenuta mediante l'implementazione del seguente processo ciclico, iterativo e convergente, grazie al supporto offerto dalla Piattaforma FAD&T. Il livello System Modeling viene segmentato in tante componenti elementari, ognuna corrispondente ad una specifica funzionalità base del sistema embedded da sviluppare (fig. 6b). In caso di esito negativo del testing, sul segmento di modello ottenuto viene prodotta una nuova versione del modello, con l'obiettivo di rimuovere la precedente anomalia, altrimenti si cicla tante volte fino ad ottenere il modello desiderato. Modellato correttamente il comportamento di una funzionalità, si può quindi procedere contemporaneamente sia allo sviluppo e al testing del firmware relativo al segmento in questione (spostandosi verticalmente verso il livello Firmware Design and Development), sia alla modellazione di un nuovo segmento a livello System Modeling (spostandosi orizzontalmente). Allo stesso modo, iterativamente e ciclicamente, anche all'interno del livello Firmware Design and Development si può procedere contemporaneamente sia all'integrazione e al testing del firmware all'interno dell' FPGA (spostandosi verticalmente verso il livello System Integration), sia allo sviluppo del firmware di un nuovo segmento (spostandosi orizzontalmente).

In generale, per ogni dato segmento di un livello, se l'attività di testing relativa produce esito negativo, allora si rieseguono, ripartendo dal livello più alto, tutte le attività relative ai segmenti coinvolti dalle modifiche apportate per risolvere le anomalie riscontrate. La specificazione dettagliata delle attività relative ad ogni singolo livello ed al modo con cui la Piattaforma FAD&T è di ausilio all'implementazione di tale processo sarà esplicitata nei successivi paragrafi. Nei sistemi embedded al momento sviluppati da T&T srl, si rileva che con tale approccio (verticale-parallelizzato, implementato su un'unica piattaforma), rispetto ad un approccio tradizionale (orizzontale-sequenziale, implementato mediante l'ausilio di più piattaforme eterogenee), si ottiene una riduzione dei tempi di sviluppo che va dal 25% al 50%. Tale risultato è in linea con quanto rilevabile in letteratura quando sono usate tecniche di ingegnerizzazione basate su modello (Model Driven Engineering).

La Piattaforma FAD&T, a sua volta, è un prodotto software e hardware sviluppato nel rispetto dei più stringenti standard di sicurezza vigenti in ambito industriale[5], secondo un processo ciclico iterativo convergente (*V-model*). Tale processo definisce le attività valide per l'intero ciclo di vita del prodotto FAD&T.

Mediante l'implementazione del *V-model,* tale piattaforma sarà in grado di fornire, a differenza degli attuali e più comuni prodotti commerciali, firmware certificabile se-

The vertical and parallel development activity is obtained by means of the implementation of the following cyclic, iterative and convergent process using FAD&T Platform. The System Modelling level is segmented in many elementary components each one related to a specific and basic functionality of the embedded system to be developed (fig. 6b). If the testing result of the segment is negative, a new version of the model will be produced, to remove the previous anomaly, otherwise this cyclic approach must be repeated until obtaining the expected model. When the model of the segment is rightly completed, it is possible to proceed at same time both on firmware development and testing of the previous segment (vertically proceeding to Firmware Design and Development Level), and modelling of the new segment at System Modelling level (horizontally proceeding). So iteratively and cyclically at Firmware Design and Development level it is possible to proceed both on firmware integration and testing (vertically proceeding to System Integration Level) on FPGA and of firmware development of another segment (horizontally proceeding).

Generally, for each segment of a level, if testing result is negative, all previous activities, starting from upper levels related to involved segments, are repeated to solve the detected anomalies. The detailed explanation of the activities for each level, and how FAD&T Platform helps to implement them, will be presented in next paragraphs. For the embedded systems developed by T&T srl it is possible to detect that with this innovative development approach (vertical-parallel, fully implemented on one platform), than a traditional approach (horizontal-sequential, implemented on several heterogeneous platforms), the time reduction is within the range of 25% to 50%. This result can also be found in literature when engineering techniques based on models are used (Model Driven Engineering).

In turn FAD&T Platform is a software and hardware product developed respecting the most restrictive safety standards of the industrial environment[5], according to a convergent iterative cyclic process (*V-Model*). This process defines the life-cycle management activities used for the FAD&T product.

Through the implementation of the *V-Model,* this platform will provide certifiable firmware according to the foreseen safety levels (*SIL - Safety Integrity Levels*) differently to the actual and more common consumer products. The required SIL is obtained also thanks to automated implementation, in the firmware generated by FAD&T, of *Fault Prevention* and *Fault Tolerance* measures, capable of ensuring the necessary reliability levels to the application produced in relation to specific Industry usage. Moreover, FAD&T prevents techniques of "hardware diversity" today used, to avoid hardware redundancy and

[5] Il settore ferroviario è assunto come settore industriale di riferimento per l'implementazione del processo di gestione del ciclo di vita del prodotto.

[5] Railway is assumed as the industrial area reference for the implementation of the product life cycle management.

condo i livelli di sicurezza previsti (*SIL - Safety Integrity Levels*). Il SIL desiderato si raggiunge anche grazie all'implementazione automatizzata, nel firmware prodotto da FAD&T, di misure di *Fault Prevention* e di *Fault Tolerance*, in grado di garantire i livelli di fidatezza necessari all'applicazione prodotta, in funzione del settore industriale d'impiego. Inoltre, a parità di fidatezza raggiungibile, e senza duplicazione dell'hardware, grazie all'impiego di FAD&T si previene l'impiego di tecniche di *hardware diversity*, attualmente utilizzate per il raggiungimento del SIL richiesto che, per applicazioni di larga scala, comportano costi elevati a causa della duplicazione dei componenti di controllo che contemporaneamente devono operare.

4. Il processo di sviluppo per il firmware dei sistemi embedded implementato grazie alla FAD&T Platform

La Piattaforma FAD&T è un applicativo software configurabile che è in grado di interfacciarsi, mediante opportuno hardware, con i sistemi embedded basati su FPGA da programmare. Essa consente di supportare i progettisti del firmware, in maniera guidata mediante interfaccia utente, nell'implementazione di un definito processo di sviluppo che si espleta nelle fasi indicate in fig. 7.

Ogni livello del processo di sviluppo del firmware, descritto in § 3, è suddiviso nelle fasi indicate in fig. 7, per ognuna delle quali sono di seguito definite le attività svolte, gli ingressi utilizzati e i risultati parziali e finali prodotti da ciascuna, grazie all'ausilio della Piattaforma FAD&T. Di ausilio alla comprensione del processo di sperimentazione sono state riportate successivamente diverse figure all'interno delle quali risultano rispettate le seguenti convenzioni:

- *Ingressi*. Scelte utente selezionate mediante l'interfaccia di FAD&T. Sono indicati mediante frecce a punti di colore blu.
- *Evoluzione del processo*. Il flusso delle elaborazioni è rappresentato mediante frecce continue di colore blu.
- *Azioni Elaborative*. Sono indicate mediante rettangoli colorati in funzione del livello di

to obtain the desired SIL that for large-scale applications are very expensive due to the hardware redundancy of the control hardware with the same dependability.

4. The development process for firmware of the embedded systems implemented by FAD&T platform

The FAD&T Platform is a configurable software application capable of interacting with the embedded systems based on FPGA to be programmed using appropriate hardware. It helps firmware designers, using a user interface in a guided manner, during the implementation of the defined development process composed by several phases as shown in fig. 7.

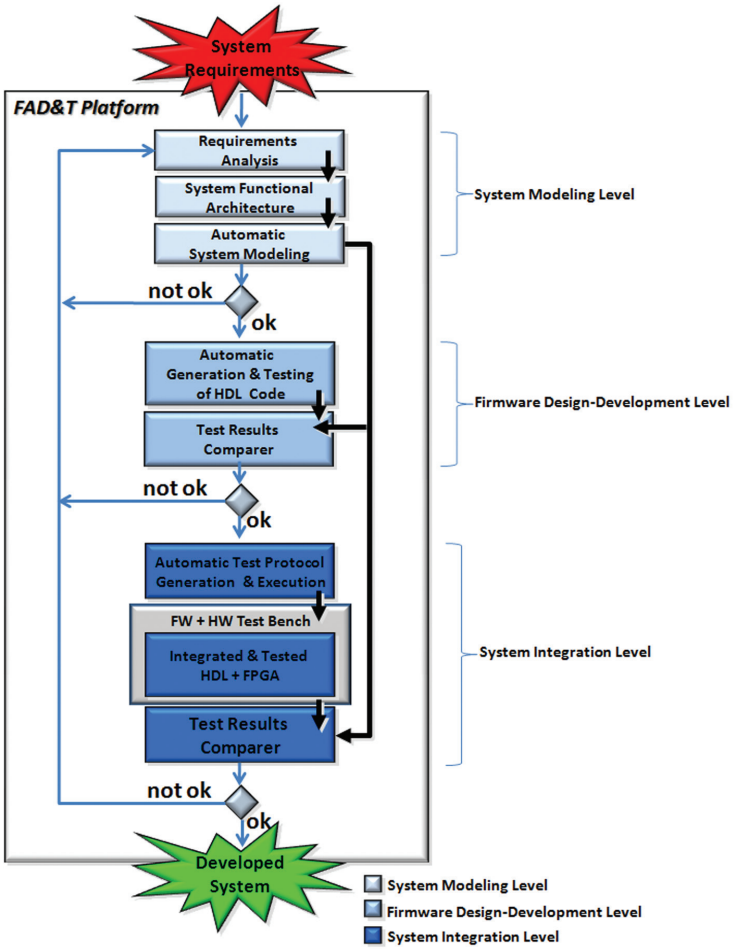

Fig. 7 - Fasi del processo di sviluppo del firmware implementate da FAD&T. *Phases of the firmware development process implemented by FAD&T (FW=Firmware, HW=Hardware).*

appartenenza, coerentemente a quanto già indicato nelle figg. 6 e 7.

- *Risultati sperimentali prodotti*. Sono riportati mediante la visualizzazione del contenuto dei file di output e risultano indicati mediante frecce tratteggiate blu.

A partire dai Requisiti di Sistema, espressi in linguaggio naturale, viene effettuata la loro analisi (*Requirements Analysis Phase*) che conduce ad una forma tale da poter essere impiegabile per la definizione dell'Architettura Funzionale del Sistema (*System Functional Architecture Phase*), prodotta sulla base di appropriate e formalizzate regole di rappresentazione gerarchizzata (fig. 8).

Successivamente, viene automaticamente generato il modello di sistema che implementa l'architettura precedentemente definita (*Automatic System Modeling Phase*). Tale modello complessivo è ottenuto dall'insieme dei modelli di base relativi alle funzionalità elementari del sistema embedded, che rappresentano i "segmenti" del livello System Modeling (fig. 6b), ad ognuno dei quali risulta applicato il processo ciclico, iterativo e convergente precedentemente descritto (§ 3). Le funzionalità elementari, sono state modellate e testate automaticamente, mediante formalizzazione con macchine a stati finiti. Il modello complessivo, viene infine automaticamente testato per

Each level of the firmware development process described in § 3 is divided in the phases shown in fig. 7. The activities of each phase are defined below using inputs and final and partial outputs obtained by means of FAD&T Platform. Some pictures have been added also to better understand this experimental process. In these pictures the following conventions have been used:

- *Inputs*. User selections by means of FAD&T interface. These inputs are indicated with blue dotted arrows.
- *Process evolution*. Elaboration flow is represented by means of blue arrows.
- *Actions*. They are indicated by means of coloured rectangles according to the related level, as already indicated in figures 6 and 7.
- *Experimental Outputs*. They are indicated using the output file visualisation. These outputs are represented with dashed blue arrows.

Starting from System Requirements analysis (*Requirements Analysis Phase*), that leads to a form usable for the definition of Functional Architecture of the System (*System Functional Architecture Phase*) on the basis of appropriate and formalized rules of hierarchical representation (fig. 8).

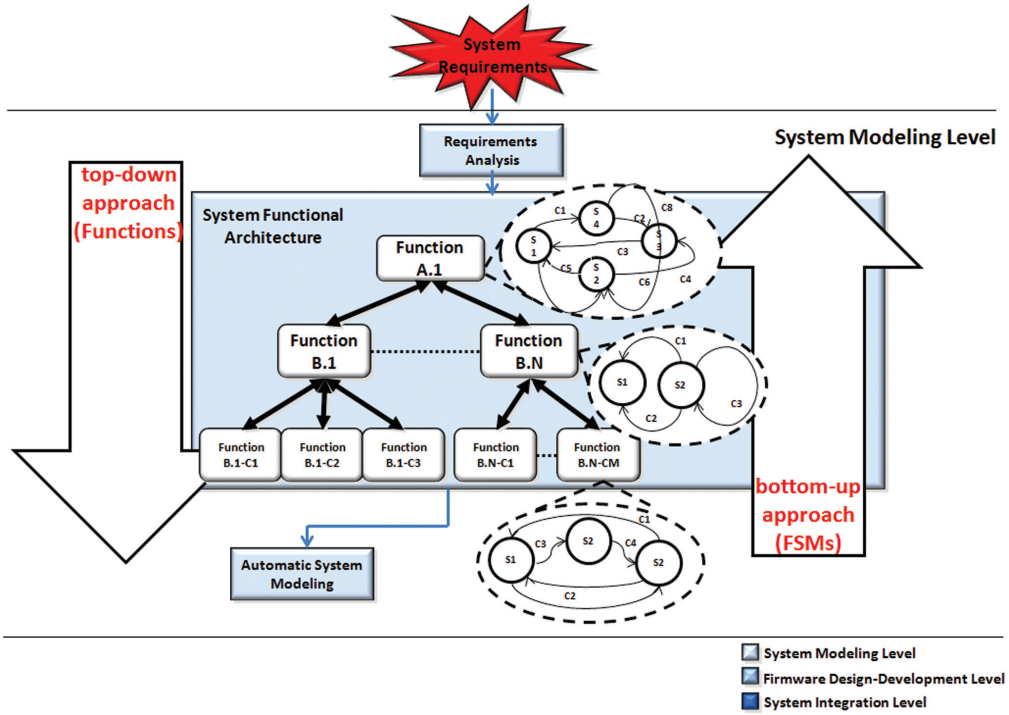

Fig. 8 - Esplicitazione del System Modeling Level implementato con l'ausilio della Piattaforma FAD&T. *System Modeling Level implementation using FAD&T Platform (FSM=Finite State Machine).*

verificare che gli output prodotti e opportunamente formalizzati, corrispondenti ai possibili ingressi, siano coerenti con quanto atteso (fig. 9). Qualora ciò non fosse, sulla base del System Modeling Test Report generato, sarà necessario:

- rieseguire la *Requirements Analysis Phase*, se il problema riscontrato fosse attribuibile ad un errore di interpretazione dei requisiti;
- rieseguire la *System Functional Architecture Phase*, qualora ci fossero impatti sulla formalizzazione dell'architettura funzionale derivanti dalle modifiche introdotte dal precedente punto e/o il problema riscontrato fosse direttamente attribuibile alla soluzione architetturale inizialmente prodotta;
- rieseguire l'*Automatic System Modeling Phase*, qualora ci fossero impatti sul modello derivanti dalle modifiche introdotte dai precedenti punti e/o il problema riscontrato fosse direttamente attribuibile al modello di sistema inizialmente prodotto.

Grazie a tale modello di sistema, così ottenuto e opportunamente testato, viene poi (fig. 10) automaticamente generato il firmware che sarà a sua volta testato (*Automatic Generation & Testing of HDL Code Phase*) per verificare che gli output opportunamente formalizzati siano coerenti con quelli attesi, e identici a quelli prodotti dall'ultimo testing del modello effettuato in precedenza (*Test*

The system model, that implements the previous functional architecture, is then generated automatically (*Automatic System Modeling Phase*). The whole model is obtained integrating the models coming from elementary functionalities of the embedded system. Each elementary model represents a segment of the System Modelling Level (fig. 6b). At each segment is applied the cyclic, iterative and convergent process previously described (§ 3). The elementary functionalities are modelled and tested automatically using formalized finite state machines. The whole model is at last automatically tested to verify if the formalized results are compliant with the values expected for each input (fig. 9). If this comparison has "negative" results, using the generated System Modelling Test Report, it will be necessary to:

- carry out again the *Requirements Analysis Phase*, if the negative results are related to a wrong interpretation of the system requirements;
- carry out again the *System Functional Architecture Phase* if there are impacts on the functional architecture coming from the previous item or directly related to "negative" results;
- carry out again the *Automatic System Modelling Phase*, if there are impacts on the system model coming from the previous items or directly related to "negative" results.

Using the system model obtained and appropriately tested, the firmware is automatically generated (fig. 10) and tested (*Automatic Generation & Testing of HDL Code*

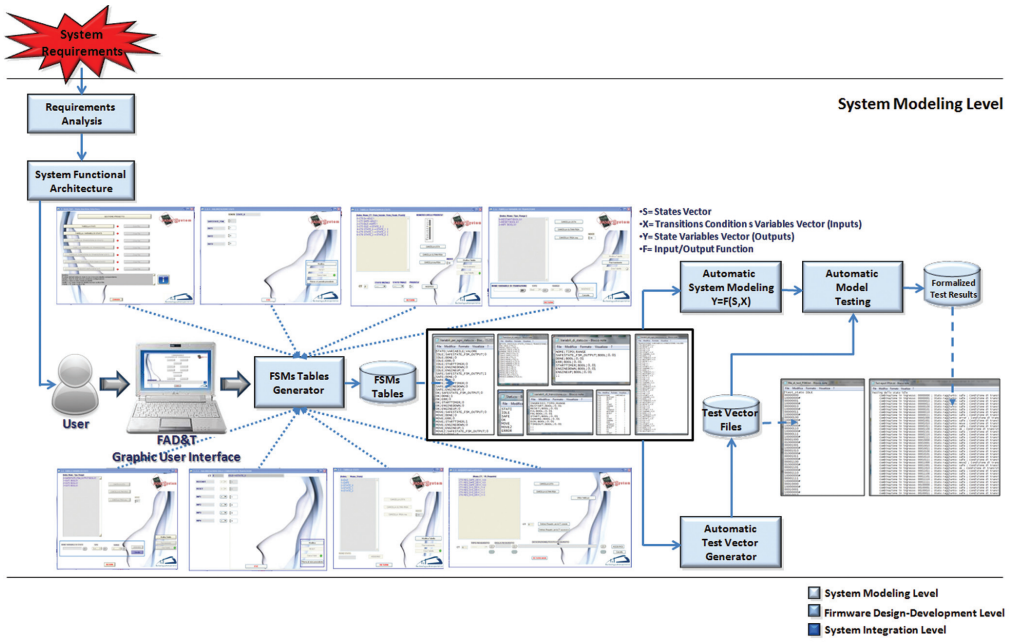

Fig. 9 - Esplicitazione del processo e dei risultati sperimentali dell'Automatic System Modeling Phase implementata con l'ausilio della Piattaforma FAD&T. *Process and experimental results explaination of the Automatic System Modeling Phase implemented using FAD&T Platform.*

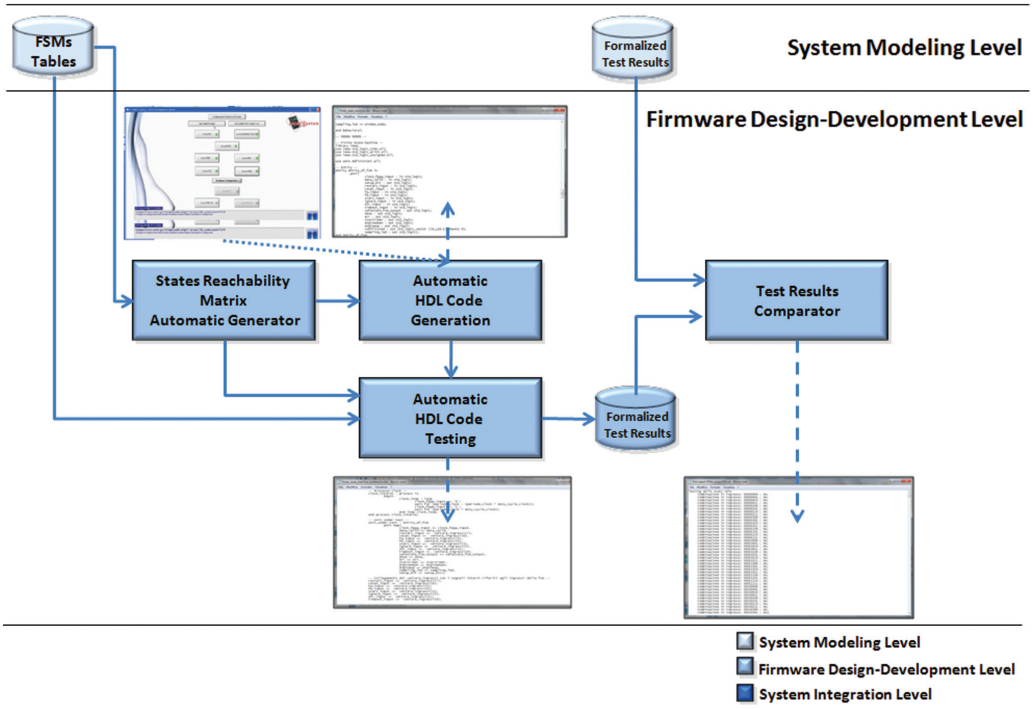

Fig. 10 - Esplicitazione del processo e dei risultati del Firmware Design-Development Level implementato con l'ausilio della Piattaforma FAD&T. *Process and results explaination of the Firmware Design-Development Level implementation using FAD&T Platform.*

Result Comparer Phase). In caso di discordanze tutte le fasi precedenti a quella corrente sono ripetute con l'obiettivo di rimuovere la causa dell'anomalia riscontrata.

Infine, il firmware correttamente testato viene integrato sull'hardware (*Embedded System*), grazie all'interfaccia hardware configurabile presente all'interno della Piattaforma FAD&T. Tale interfaccia è in grado di essere impiegabile verso i sistemi embedded forniti attualmente dalle principali case produttrici. Viene automaticamente generato, e successivamente eseguito, il protocollo di test che ne verifica l'integrazione firmware+hardware (*Automatic Test Protocol Generation & Execution Phase*), secondo quanto indicato in fig. 11. Anche per questa fase, qualora vengano riscontrate discordanze tra risultati attesi e risultati ottenuti (*Test Result Comparer Phase*), comparati anche rispetto agli output formalizzati prodotti durante il testing del solo modello del sistema (Gli output formalizzati sono uguali per costruzione a quelli prodotti alla fine della generazione del firmware), allora tutte le fasi precedenti a quella corrente sono ripetute al fine di convergere ciclicamente verso il sistema definitivamente sviluppato, testato e rilasciabile. Tutti i cicli necessari potranno essere rapidamente svolti grazie all'approccio automatizzato usato dalla piattaforma FAD&T.

Phase). The formalized results of these testing activities can be compared with the formalized results coming from the System Modeling Phase (*Test Result Comparer Phase*). If the comparison detects anomalies, then all the previous phases must be repeated to remove all the problems.

Finally, the correctly tested firmware is integrated in the hardware (*Embedded System*) by means of the configurable hardware interface of the FAD&T Platform. This interface can be used for the most important embedded systems currently produced. Finally the test protocol is generated and executed (*Automatic Test Protocol Generation & Execution Phase*) to verify firmware+hardware integration, as shown in fig. 11. After this phase, it is possible to automatically compare (*Test Result Comparer Phase*) the last formalized results with the formalized results coming from the System Modeling Phase (These formalized results are also the same as those coming from automatic firmware generation). If the comparison detects anomalies, then all the previous phases must be repeated to remove all the problems in a convergent cyclic process. Otherwise, the testing of the integrated firmware+hardware is completed and the dependable embedded system can be released as developed system. All necessary cycles can be quickly done thanks to the automatized approach used by FAD&T Platform.

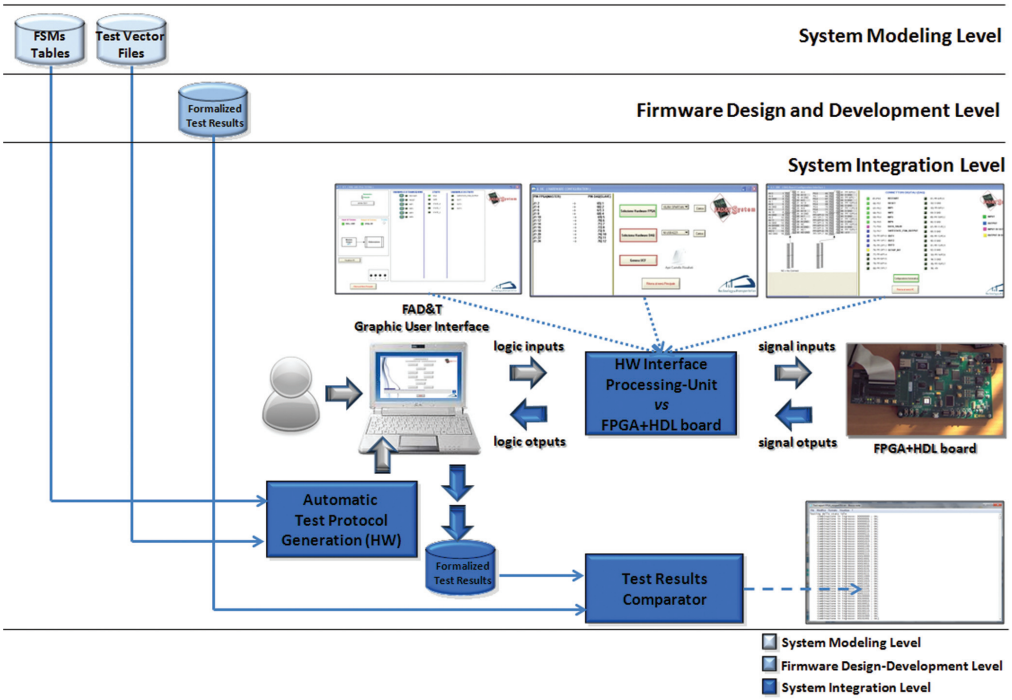

Fig. 11 - Esplicitazione del processo e dei risultati del System Integration Level implementato con l'ausilio della Piattaforma FAD&T.
Process and results explaination of the System Integration Level Level implementation using FAD&T Platform.

5. Le funzionalità della Piattaforma FAD&T e i suoi possibili contesti applicativi

La Piattaforma FAD&T implementa le seguenti funzionalità:

- Produzione Architettura Funzionale Formalizzata del sistema *embedded* da programmare;
- Generazione Automatica del Modello di Sistema;
- Generazione Automatica dei Vettori di Test;
- Generazione automatica del Protocollo di Test per il Modello di Sistema;
- Generazione Automatica di *firmware* certificabile (SIL X[6]) per applicazioni *safety & mission critical*;
- Generazione Automatica di Protocolli di Test *firmware*;
- Esecuzione Automatizzata di Protocolli di Test *firmware*;
- Generazione Automatica di Protocolli di Test *firmware* e *hardware* integrati;
- Esecuzione Automatizzata di Protocolli di Test *firmware* e *hardware* integrati;

[6] "X" è il livello di integrità della sicurezza da raggiungere da specifica, e varia nell'intervallo 0-4.

5. The FAD&T Platform functionalities and its usage context

The FAD&T Platform implements the following functionalities:

- Formalized Functional Architecture of the Embedded System to be realized;
- System Modelling Automatic Generation;
- Test Vectors Automatic Generation;
- Test Protocols Automatic Generation for System Model;
- Automatic Generation of certifiable *firmware* (SIL X[6]) for *safety and mission critical* applications;
- *Firmware* Test Protocols Automatic Generation;
- *Firmware* Test Protocols Automatic Execution;
- Test Protocols Automatic Generation for integrated *firmware+hardware*;
- Test Protocols Automatic Execution for integrated *firmware+hardware*;

[6] X" is the safety integrity level to be implemented from requirements and it is a number in the range 0-4.

- Generazione Automatica di Test Results formalizzati;
- Comparazione Automatizzata tra Test Results formalizzati.

La piattaforma FAD&T è impiegabile per le seguenti applicazioni:

- Ausilio alla Progettazione e allo Sviluppo di firmware fidato per FPGA (SIL X);
- Testing Automatizzato di firmware integrato su FPGA (SIL X);
- Validazione di Sistemi Safety & Mission Critical basati su FPGA;
- Ausilio alle attività di Assessment di "parte terza" rispetto ai Produttori di Sistemi Embedded basati su FPGA.

Ultimata la sperimentazione, T&T sta preparando la documentazione, relativa alle attività di processo e di prodotto, per la certificazione CENELEC della Piattaforma FAD&T.

- Formalized Test Results Automatic Generation;
- Formalized Test Results Automatic Comparison.

The FAD&T Platform can be used in the following applications:

- Design and Development of Dependable Firmware for FPGA (SIL X);
- Automated Testing of integrated Firmware on FPGA (SIL X);
- Validation of Safety and Mission Critical Systems based on FPGA;
- Third party Assessment activities for Safety Certification of Embedded Systems Manufacturers based on FPGA.

Once the experimentation is ended, T&T is preparing the documentation related both to process and product to start the CENELC certification of the FAD&T Platform.

BIBLIOGRAFIA – *REFERENCES*

[1] CEI EN 50126:2006, *"Railway applications. The specification and demonstration of reliability, availability, maintainability and safety (RAMS)"*.

[2] CEI EN 50128:2002, *"Railway applications. Communications, signalling and processing systems. Software for railway control and protection systems"*.

[3] CEI EN 50129:2004, *"Railway applications. Communication, signalling and processing systems. Safety related electronic systems for signalling"*.

[4] CEI EN 61508:2002, *"Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems"*.

[5] C. Brandolese, W. Fornaciari, *"Embedded systems. Hardware and software development for dedicated systems"*, Pearson Prentice Hall, 2007.

[6] M. Zwolinsky, *"VHDL. Digital systems design"*, Pearson Prentice Hall, 2007.

[7] I. Sommerville, *"Software engineering"*, Pearson Addison Wesley, 2005.

[8] C. Ghezzi, M. Jazayeri, D. Mandrioli, *"Software engineering. Fundamentals and principles"*, Pearson Prentice Hall, 2004.

[9] M. Fowler, *"UML Distilled"*, Pearson Addison Wesley, 2004.

[10] L. Vetti Tagliati, *"UML and software engineering"*, Tecniche Nuove, 2003.

[11] A. Di Febbraro, A. Giua, *"Discrete events systems"*, McGraw-Hill, 2002.

[12] A. Gill, *"Introduction to the theory of finite state machines"*, McGraw-Hill, 1962.

[13] C. Batini, L. Carlucci Aiello, M. Lenzerini, A. Marchetti Spaccamela, A. Miola, *"Programming Fundamentals"*, Franco Angeli 1992.

[14] M. Malatesta, *"C Programming language"*, Città Studi De Agostini, 2007.

[15] B.S. Dillon, *"Engineering plus safety"*, World Scientific, 2003.

[16] A. D'Auria, R. Nappi, E. Napoli, *"FPGA design for a railway routing system"*, University of Naples "Federico II" Electronic Engineering Faculty, 2009.

[17] C. Di Chiara, R. Nappi, E. Napoli, *"VHDL development and testing on FPGA for safety critical applications"*, University of Naples "Federico II" Electronic Engineering Faculty, 2010.

[18] F. Grasso, R. Nappi, E. Napoli, *"Analysis, definition and development of VHDL programming rules for safety critical applications"*, University of Naples "Federico II" Electronic Engineering Faculty, 2011.

[19] J.C. Laprie, *"Dependable Computing: Concepts, Limits, Challenges"*, LAAS-CNRS, 1995

[20] P. Mohagheghi, V. Dehlen, *"A Review of Experiences from Applying MDE in Industry"*, SINTEF, 2008.