Pre-silicon Digital Functional Verification Engineer (FVE)

Job Description

Table of Contents

This document provides a technical overview of the job requirements for Pre-Silicon Digital Functional Verification Engineer (FVE) positions.

Looking for a Good Start to Your Career?

This document is aimed at students, fresh graduates and those looking to embark on a new career path. It contains the following:

- a guide to entering the job market (yes, there is life after university :))
- an insight into the company's expectations in terms of skills and knowledge
- information that will put you in a better position to take decisions about your career

Your Chance to Be Part of Something Big

What exactly are FVEs? Who needs them and why? Read on and you will learn which industries are crying out for FVEs, what kind of companies employ FVEs, some alternative names for FVEs and what it takes to become one.

Industries:

As an FVE you will be employed by a company that is part either of the Semiconductor Industry or the Electronics Industry. These are the industries responsible for the Internet of Things, Cloud Computing, high-performance computing, telecommunications, drones, remote surgery, Mars exploration, autonomous cars, wearables, etc. If you are able to work from a bench in the middle of a park, it is because you use a mobile device that contains a high performance, battery-friendly IC (Integrated Circuit). You get the idea… anything that has an IC also requires verification.

Companies that employ FVEs:

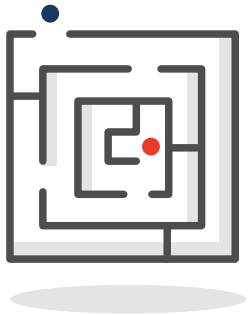
- ASIC producers (Intel, Apple, Infineon)
- FPGA producers (Xilinx, Altera, Atmel)
- Semiconductor factories (Samsung, Intel, TSMC, GLOBALFOUNDRIES)
- Electronics industry companies (Sony, nVidia, Samsung)
- Telecommunications companies (Ericsson, Nokia, Cisco)
- [Verification] IP suppliers (MoreThan IP, Imagination)
- Electronic Design Automation Companies (Mentor, Cadence, Synopsys, AMIQ EDA)
- Consulting companies that provide verification services (AMIQ Consulting, Verilab)
- Training services companies (Doulos)

Other names for FVE:

If you search for an FVE position, you will also find it under one (or a combination) of these tags:

- Digital/Analog/Mixed Signal Verification Engineer
- ASIC or FPGA Verification Engineer
- Functional [Hardware] Verification Engineer
- Pre-Silicon Verification Engineer
- Hardware Verification Engineer
- RTL Verification Engineer

Required Steps to Become an Expert



Studies:

Degree courses that develop the required mix of knowledge and skills to become an FVE are: Electronics/Telecommunications, Computer Science, Automatic Control Systems, Information Technology and Mathematics.

I am a junior employee. Now what?

You should focus on developing your verification skills and increasing your knowledge of different architectures, devices, communication protocols, tools, etc. As an FVE you will work with an interesting array of architectures (e.g. multi-core, Network-on-Chip), IPs (ARM, DDR4-LP), devices (communication switches), communication protocols (Ethernet, RapidIO, USB), tools (simulators, fault injection, formal engines), programming languages, methodologies and platforms (FPGA, emulation). If you have analog design knowledge and have mixed feelings about digital verification, you can opt to do mixed signal verification, which provides a nice blend of both fields. You can also switch back and forth between simulation/emulation-based verification and FPGA/final product verification in lab.

You will be part of a team and a working environment that will allow you to grow your communication and leadership skills. Frequent verbal or written communication will improve your English speaking and writing skills. You will also be able to improve your leadership skills by seizing the opportunity to become a team leader. As part of a multicultural team you will gain first-hand experience of other cultures, meaning you will also see growth on a personal level.

And as you gain more experience and knowledge, you will take on higher responsibility roles, such as system level verification lead or verification department technical lead.

What if I change my mind?

An FVE position offers you the flexibility to change your career path at a later date, given that you will already be familiar with a mix of software and hardware technologies, your will have used OO-programming and software design patterns on a daily basis, and will have experience of working in a multicultural environment, etc. There are a number of natural choices available to you if you want to work in the semiconductor and electronics industries, e.g. digital design engineer, embedded systems software/firmware developer or software developer for an EDA company. And you will always have the option to switch to mobile/web/desktop application development.

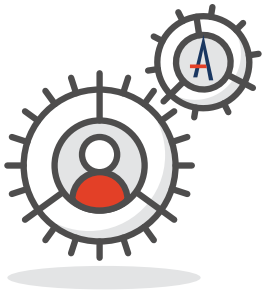
A Pool of Opportunities

Pre-silicon digital functional verification is a niche area within the field of hardware development. It is smaller in size (in terms of number of companies, potential jobs and available specialists) than the web/mobile/desktop software development industry. But there is still a high demand for specialists, especially in Europe, even when taking this difference in size into account.

The majority of companies that hire FVEs have their offices in Europe (mostly Germany, France, Italy, Sweden, Norway, Denmark, Finland and Holland), UK, USA, Israel and Asia (India, China, Singapore, Taiwan, Japan). You will be employed either directly by one of these companies, or indirectly through a consulting firm like AMIQ Consulting.

If you are eager to travel or live abroad, you will easily find a contract that suits your needs, either as an employee of a consulting firm or as an independent contractor. Most of the time you will negotiate the amount of on-site time. Outsourcing functional verification is a growing trend among IC producers, meaning you will also find FVE jobs in locations other than those mentioned here.

What is the Role of an FVE?



The role of an FVE role is to identify functional bugs before the IC production process begins. Your work will help the chip maker avoid production respins, recalls, brand damage, etc. FVEs are essential to making ICs better and safer.

The cost per bug fix in ICs is much higher than the cost per bug fix in software. And the same is true of turnaround times. You can fix software bugs in days or minutes and then send out a patch to your users with almost no impact on the usability of the software.

Functional hardware bugs are easy to fix if they are discovered before IC production begins (i.e. during the pre-silicon phase), in which case the effort involved in fixing them is comparable to that of fixing software bugs. However, functional hardware bugs that make it to the production stage will be present in all ICs and may impact all those who use them. In the worst case scenario, these bugs will only be eradicated if the IC maker goes back to the pre-silicon phase, fixes the bug and restarts production (i.e. a respin). What's more, any recall costs are to be added to the respin costs, i.e. when the IC maker has to recall buggy ICs and replace them with new ones. You can read about two famous examples of hardware bugs and their implications here (at a cost of $1 billion!, there is an in-depth explanation here) and here ($475 million). And you can find out more about the costs involved in IC production here and here. Besides cost, there are also safety-related issues that any FVE must be aware of. Medical and military equipment, aeronautical devices, automotive and space-grade applications, to name just a few, are not allowed to contain any functional bugs. Why? Because human life is at stake. You can learn more about the importance of safety-critical system verification on Yoav Hollander's blog. Last but not least, FVEs are also responsible for finding specification bugs. Implementing functionality based on a buggy specification will only lead to a buggy IC.

You will find a detailed account of the importance of functional verification and the latest trends in the field in the Wilson Research Group and Mentor Graphics study.

What Does an FVE Do?

(Mostly software, with a dash of digital design and a pinch of verification planning.
All topped off with a generous helping of communication.)

The following section contains a detailed description of the various tasks you will undertake.

- **Study the specification of the Device Under Test (DUT)**

You must understand what functions the verified module is supposed to perform, how it interacts with other devices in the system, what the weak and strong points of its implementation are, and what its legal or illegal configurations are. Once you understand the specification and have clarified any issues or questions, you will move on to the verification planning phase. You can learn more about this process in another post of mine: How to Read a Specification.

- **Write the verification plan document**

During this stage you need to write the verification plan document, which defines what features are to be verified and specifies the checkers and metrics (e.g. functional coverage, code coverage, assertion coverage) required to prove the features have been verified.

Besides writing the verification plan, you also need to write a verification specification document containing details of the verification strategy. The verification strategy describes the mix of tools (e.g. simulators, verification automation tools, version control systems, build systems), methodologies (functional verification

formal verification, directed testing, assertion-based verification) and metrics (functional coverage, code coverage) you are going to use to achieve the verification goals established in the verification plan. The verification specification also contains a description of the verification environment architecture (e.g. structure, configuration, verification component layering), the data flows through the verification environment, and the reference implementation (e.g. Matlab model, SystemC/C/C++ model, SystemVerilog model).

Verification planning is a process that can be done either in one step (i.e. for simple DUTs) or over a succession of verification planning sessions.

- **Implement the Verification Environment**

As soon as the verification planning is done you will proceed to implement the verification environment using hardware verification languages (e.g. SystemVerilog, e-language) and hardware verification methodologies (UVM, UVM-e). This involves coding new verification components, integrating existing verification components, reference models, implementing metrics collectors and data/timing checks, integrating the DUT's RTL source with the verification environment, creating the required build infrastructure, and implementing scenarios and tests.

This phase is mostly about programming using object/aspect-oriented languages (e.g. SystemVerilog, e-Language) and software design patterns. I would say that, as a rule, pre-silicon functional verification is 90% object-/aspect-oriented programming and 10% digital circuit design.

- **Simulate the design, debug**

Once you have finished coding a self-contained portion of the verification environment you will begin running simulations using an RTL simulator (e.g. Incisive by Cadence, QuestaSim by Mentor, VCS by Synopsys). Any bugs will emerge either in the design or the verification environment. These will have to be fixed together with the designer, system architect and software engineers.

- **Collect metrics and track verification progress**

During this phase you will run test suites in order to collect metrics and achieve verification goals. To do this you will use verification automation tools, such as Cadence's vManager or Mentor's Questa Verification Manager. Any metrics collected will be analyzed and eventually the verification environment will be enhanced to address them.

- **Report bugs**

Once you identify a bug or an issue, or if you have doubts as to the correctness of the design/specification, you should open a discussion with one of the DUT's stakeholders (e.g. the designer, architect, software engineer). You can do this either informally, over a cup of coffee or by email, or formally, by means of a bug reporting system (e.g. Bugzilla, YouTrack, Jira Software) .

- **Collaborate with others**

Collaboration skills are highly valued, just like in any other high-tech job. You will always be part of a team, be this a team of two, ten or more people. You will take part in various meetings and bug analysis sessions, write emails, write documents (e.g. specifications, reports), give presentations, train others, hand over your work to others, and take over work from others, etc. All of these activities will help improve your communication and language skills. Multicultural verification teams mostly work in English, but other languages, like German, Hebrew and French, can be a plus.

Required Practical Skills

- **Practical experience with Generic and Object Oriented Programming**

You will need generic programming experience of languages like C. The reason for this is that all other programming knowledge and skills have generic programming as a basis.

Practical experience of languages like Java and C++ is a must. All verification domain specific languages have object-oriented traits and 80% or more of functional verification is programming.

You need to understand concepts like inheritance, namespaces, and encapsulation. And by practical experience I mean "complete an average complexity project from start to finish involving activities such as coding, debugging and testing".

- **Practical experience in Software Debug**

Debugging the verification environment or the DUT's RTL can be time consuming if you have no experience of software debugging (e.g. using GDB for C/C++, Eclipse for Java). Simple debugging tactics like messaging, step-by-step debugging, application profiling should be second nature to you.

- **Practical experience with Verilog/VHDL**

You should have some digital circuit design experience of either Verilog or VHDL. You will be required to understand and debug components such as FIFOs, finite state machines, pipelines, communication protocol interfaces, memories, clock domain crossings, etc. You can use Asic World and Eda Playground in order to practice these skills.

- **Practical Experience with Linux environment**

The semiconductor industry uses Linux-based OSes extensively for research & development activities, so experience of Linux is highly appreciated. Verification projects have lots of similarities with software projects. This is why experience of version control systems (e.g. GIT, SVN), build systems (Make-, Perl- or Python-based), and [shell] scripting languages (Perl, Python, Bash) will allow you to adapt quickly to different project infrastructures.

Required Knowledge

This section details the theoretical knowledge you should have before embarking on a career in functional verification.

- **Algorithms and Data Structures**

You should know the basics of algorithm implementation and data structure design. Data checking and protocol monitoring require knowledge of various search algorithms, while layered communication protocols require knowledge of data structure implementations and transformations, etc.

- **Digital Circuits Design**

You need to master subjects such as Boolean algebra, combinational logic, sequential logic, latches, flip-flops, and finite state machines in order to understand and debug digital circuits. You also need to understand their real life applications, such as FIFOs, muxes, look-up tables, clock domain crossings, memories, simple serial or parallel communication protocols (e.g. I2C, SPI, AHB), and error correction (de)coders. Knowledge of basic design pattern implementation in Verilog or VHDL is a plus.

Nice to Have

The knowledge and experience described in this section will make a big difference since they will a) help you understand the system you are verifying at a higher level of abstraction, b) help you verify various types of functionality, and c) allow you to move through different integration levels and identify bugs at an architectural level.

Some systems or higher levels of integration you may encounter will require embedded programming experience (i.e. low level programming, debugging using tools such as microcontroller/processor emulators.

It will also be considered a plus to have used at least one of the commercially available simulators (e.g. Incisive by Cadence, QuestaSim by Mentor or VCS by Synopsys) to run simulations, inspect waveforms, debug RTL issues or analyze coverage metrics (e.g. code coverage).

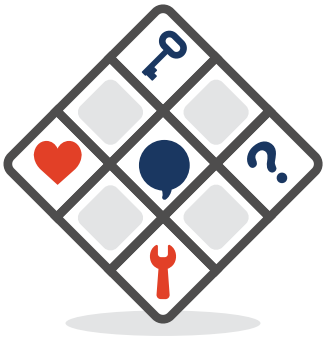
Some projects may also require experience of the FPGA workflow (synthesis flow, floorplanning, timing constraints), such as through prototyping projects or collaboration with a lab. Experience of Matlab, Octave or similar languages is a big plus if you need to verify signal processing or mathematical functions. These types of DUT require a mathematical model that is almost impossible to implement using a hardware verification language or too complex to implement using C/C++.

Digital Signal Processing knowledge will reveal to you the strengths and weaknesses of various algorithm implementations, which in turn will increase the odds of your any finding bugs. Communication Protocols and Networking knowledge will help you to understand the corner cases of a protocol you need to stress. Knowledge of Microprocessor/Microcontroller Architecture will allow you to understand the data computing flows inside the system and how they interact with other components, both hardware and software. Familiarity with System-on-Chip Architecture (e.g. the Raspberry Pi's BCM2836 SoC) and Network-on-Chip (e.g. AMBA-AHB) will enable you to understand the architecture of highly integrated devices. These are all complemented by experience and knowledge of embedded programming.

Software Design Patterns will provide you with the tools required to develop flexible, reusable and debug-friendly verification components and environments. It is also useful to have experience of Analog Design, especially if you plan to follow a mixed signal verification career path.

Required Soft Skills

There are a few soft skills that are a must and which one way or another are prerequisites for any job you might apply for:



- Determination
- Inquisitiveness
- Detail-oriented approach
- Good English communication skills, both in writing and speaking

A future post on the AMIQ blog will provide you with a more detailed account of the soft skills, perks and challenges of this job.

Further Reading and Resources

To find out more just send an email to career@amiq.com.

Other interesting resources:

- Semiconductor Industry Infographics - Infographics that help you understand the semiconductor industry at a glance
- Fabless Semiconductor Companies
- Semiconductor Fabrication Plants
- semiwiki.com – a forum dedicated to semiconductor industry news
- Details of IC production here and here
- AMIQ Consulting Blog
- Verilab Blog
- Wilson Research Group and Mentor Graphics study
- Accounts of some very expensive software bugs here ($370 million) and here ($327.6 million).
- A nice intro to Verilog, SystemVerilog on Asic World
- A commercial simulator on Eda Playground