HOCHSCHULE RAVENSBURG-WEINGARTEN UNIVERSITY OF APPLIED SCIENCES

Improving Hardware Security through Encryption of JTAG TAP Ports: An Analytical Comparison of Algorithms, with a Focus on RSA

Soham Sanjay Dekhane, Andreas Siggelkow

***Abstract*—In today's age it is vital to protect our devices from cyber threats. This research paper explores the field of hardware-based cybersecurity specifically focusing on securing Joint Test Action Group (JTAG) Test Access Ports (TAP). This research evaluates different encryption algorithms' effectiveness in enhancing the security of these ports, which are one of the most widely used debug ports for embedded systems. This research focusses on the RSA (Rivest Shamir Adleman) encryption algorithm analyzing its strengths and weaknesses compared to other cryptographic methods. Through an analysis, the aim is to offer insights into selecting the most robust and efficient encryption techniques for reinforcing JTAG TAP port's security. By analyzing and comparing encryption techniques this research provides insights, into ongoing efforts aimed at enhancing the security of embedded systems.**

***Index Terms*—RSA, JTAG, TAP, VHDL, debug, cryptography**

I. Introduction

Cyber threats present enormous risks to individuals, businesses, and even entire nations in today's connected world. Hardware security and trust have become a pressing issue during the last two decades due to the globalization of the semi-conductor supply chain and ubiquitous network connection of computing devices. Strong key management and strong authentication are provided by cryptographic hardware modules.[7] Only authorized individuals can access sensitive information and vital systems because of these hardware-based cryptographic solutions. Individuals and organizations can reduce the danger of unauthorized access, data leakage, and other criminal activity by utilizing hardware cryptography. Encryption is used in almost everyday life right from opening garage doors to credit cards. For such encryptions or cryptographic operations, generation of unique, random and secure keys is very important. Debug ports can be extremely helpful in designing embedded systems but they present a huge vulnerability. From the perspective of the security chain's resilience, random number generation is a vital task in modern encryption and security applications. In reality, the encryption keys needed for ciphers are created using random numbers. Therefore, any flaw in the key generation procedure may allow information to be leaked that might be exploited to crack even the most robust cipher [4]. A hacker with malicious intent can get physical access of the system using this debug port can cause a huge damage. The obvious solution is securing the debug port by limiting the access to it by implementing some encryption algorithm. Hardware based cybersecurity; i.e. integrating cybersecurity measures in the processing unit of the embedded system itself can provide with various advantages; the biggest being that it is not at all vulnerable to cyber attacks unless the attacker has physical access to the hardware.

Soham Sanjay Dekhane, sohamdekhane@gmail.com, Andreas Siggelkow, andreas.siggelkow@rwu.de.Hochschule Ravensburg-Weingarten, Doggenreidstraße, 88250 Weingarten.

II. JTAG

JTAG, also known as the Joint Test Action Group is an widely-used technology, in the field of digital testing. Its primary purpose is to ensure that electronic systems operate correctly and to diagnose any faults they may have. JTAG consists of components that work together to make testing processes efficient and effective. At the core of this system is the Test Access Port (TAP) which acts as a communication interface between test equipment and the on chip test logic. The TAP includes registers like the Instruction Register (IR) and Data Register (DR) which allow for data exchange during testing procedures. Through a set of predefined instructions JTAG can perform tasks such, as scan testing, memory testing and providing debug support. The JTAG is a serial communication, four wire protocol. The four signals TDI, TDO, TMS and TCLK with TSRT being an extra optional signal are a part of the JTAG protocol which belongs to the IEEE 1149.1 standard as shown in the image 1. JTAG's structure and functionalities provide a seamless flow in the testing process. The division of the TAP registers into the IR and DR, along with the availability of various test instructions, allows for a clear hierarchy and organization of operations. This ensures that testing procedures can be carried out methodically and precisely, leading to more accurate fault identification and diagnosis.

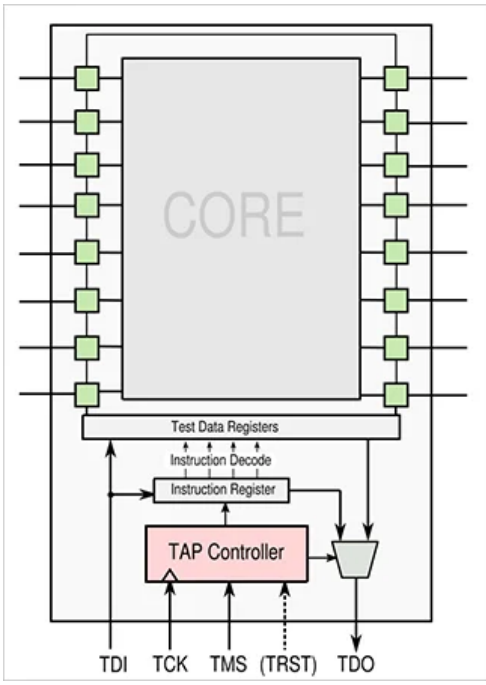


Figure 1. JTAG Registers [1].

Almost all digital systems have a debug interface [10] with different possibilities to attack the system [6]. This debug logic connects all sub-blocks in the system by means of a shadow bus system in order to test or debug it. This could act as a back door which is not secured. To equip this back door and all connection points of the debug bus with a lock, is the focus of the system introduced in the following. It is just the base system for different cipher/debug pairs, it is the base of a system evaluation.The lock can be a cipher system. Parallel to the debug problem is the update over the air possibility in such systems, especially modern cars and IoT. Also this back door can be secured by ciphering. An emulator of this kind has been presented in [5].

The back door itself is the well known JTAG port [8]. The element, which accesses all logic on chip is the JTAG port together with the test access port (TAP) controller. The TAP-Controller is implemented as a finite-state-machine (figure 2). The states of this FSM can be summarised as below.

- Test-Logic-Reset: This command resets JTAG circuits. It returns to this state whenever the TRST (optional) signal is asserted. Remember that if TMS is set to 1 for 5 consecutive TCK cycles, the TAP controller will return to this state regardless of the state it may be in. We can still reset the circuit, then, even without the TRST signal.
- Run-Test/Idle: The FSM is in this state as it waits for some test operations to finish.
- Select-DR/Scan and Select-IR/Scan: These are temporary states that enable the corresponding Register's test data sequence to be started (the selected IR is in the Select-IR/Scan state, and the selected DR is in the Select-DR/Scan state).
- Capture-DR and Capture-IR: During this state, data can be loaded simultaneously into each Register.
- Shift-DR and Shift-IR: In these states, the necessary test data is serially loaded (or unloaded) into (or from) the appropriate Register. If you look at Figure 2, the TAP controller will remain in this state as long as TMS is zero. One data bit is shifted through TDI (or TDO) and into (or out of) the chosen Register for each clock cycle.
- Exit1-DR and Exit1-IR: All parallel-loaded (from the Capture-DR and Capture-IR state) or serial-loaded (from the Shift-DR and Shift IR state) data are held in the Register in this state.
- Pause-DR and Pause-IR: The FSM pauses here in order to await an external operation.
- Exit2-DR and Exit2-IR: These states signify the conclusion of the Pause-DR or Pause-IR operation and allow the TAP controller to return to the Shift-DR or Shift-IR state for additional data to be shifted in (or shifted out).
- Update-DR and Update-IR: The test data stored in the first flop of the Register (typically all Registers have two flops for each bit; we'll talk about that later) is loaded to the second flop in this state.

The signal timing is defined as follows: The **test mode select (TMS)** will be captured with every rising edge of **test clock (TCK)**. Also **test data in (TDI)** will be taken with the rising edge of TCK. Contrary to this, **test data out (TDO)** will be driven with the falling edge of TCK. So, the wiring to a second chip, which receives the output of the actual SoC, could be allowed a delay of one half of the period of TCK.

III. Encryption vs Hashing

For handling the data during the exchange via JTAG, there are 2 possible options. The data can either be sent as a plain text or it can be sent in a form which is completely unrecognisable. Sending the data as a plain text is obviously very risky as a hacker with malicious intent can easily get access to this data and it would not remain a secret anymore. If the data is to be sent as completely unrecognisable, there are again 2 options; the data needs to be either encrypted or hashed.

A. Encryption

Encryption involves converting data into a format using an algorithm and a key. The encrypted data, also known as ciphertext can only be deciphered by someone who possesses the corresponding decryption key. Encryption is commonly used to protect data during transmission (data, in transit). When stored on devices or servers (data at rest). This means that the encryption process is a reversible one and the original text/data can be obtained by decrypting it.

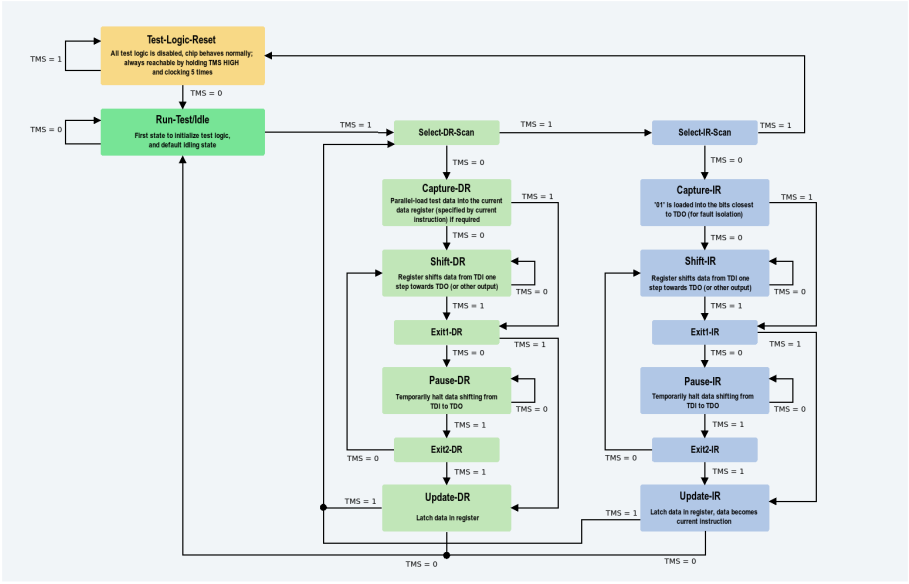


Figure 2. TAP-FSM [1].

B. Hashing

Hashing, unlike encryption functions is a one way process where data is transformed into a string of characters called a hash value. This hash value acts like a fingerprint ensuring the originality and authenticity of the data. Carefully designed hashing algorithms create collision hash values meaning it is computationally difficult to find two inputs that produce the same hash value. The property of collision resistance guarantees that if an attacker modifies the data the resulting hash value will change. This change serves as an indication that the data has been altered, notifying the recipient about information compromise. Hashing is particularly useful, for verifying file integrity when stored or transmitted over networks. By comparing the received files hash value with its counterpart one can confirm if any tampering has occurred.

When comparing Encryption and Hashing for the goal of this research work, it is clear that encryption has to be used to fulfill the aim. This is because the process of hashing, though secure, is an irreversible process and the original data can never be obtained once it is hashed. When exchanging data through JTAG, it would be computationally very heavy to generate the hashes for every permutation of the message data and compare it with the target hash. Considering an example of 16 bit hashed data being transmitted over JTAG, the worst possible case scenario would be generating 65535 different hashes and comparing them with the transmitted hash inorder to authenticate the data.

IV. Comparing various cryptography algorithms

The debug interfaces opens a back door which can be easily accessed by a hacker with a malicious intent. To secure the debug interface, some type of cybersecurity algorithm is required. But the question arises which one. A software based algorithm or a hardware based. When it comes to software-based security, it's all about using programs and applications to protect a system from malware, viruses, and other cyber threats. Many businesses and organizations rely heavily on software-based security because it's affordable and readily available. Examples of software-based security include antivirus programs, encryption software, and firewalls. While these can be effective in stopping many types of cyber threats, they still have their limitations. One major disadvantage of software-based security is that it's vulnerable to attacks from hackers who know how to exploit software vulnerabilities. No matter how sophisticated the software is, it will always be prone to vulnerabilities. Another disadvantage of software-based security is that it may not protect against physical attacks. If a hacker gains physical access to a computer or other device, software-based security will not be able to stop them. While software-based security can be effective in certain situations, it's not always the best solution. Meanwhile, hardware-based security involves using physical devices to safeguard networks, systems, and data. It offers a more robust defense mechanism against cyber threats, as compared to software-based security. Hardware-based security includes devices such as smart cards, token-based authentication, and biometric authentication. These devices provide an extra layer of security that cannot be duplicated by software. Hardware-based security is more secure than software-based security. This is because hardware devices are specifically designed to perform security functions, whereas software-based security can be compromised by vulnerabilities in

the software. Additionally, hardware-based security devices can protect against physical attacks and are inherently more difficult to hack. Before choosing a perfect cybersecurity algorithm, it has to be made sure that the correct algorithm is chosen by weighing the pros and the cons. The first decision would be whether to select a symmetric cryptography algorithm or an asymmetric one. Symmetric Cryptography employs a single key for both the encryption and decryption processes. It is just like a lock that can be unlocked using the same key used to lock it. Similarly, in symmetric cryptography, the sender and receiver share the same secret key, which is utilized to encrypt and decrypt the data. As a consequence, this method is highly efficient and suitable for securing large volumes of data. However, the main challenge lies in the secure distribution of the shared key among the communicating parties. Without proper measures, the secrecy may be compromised, leading to vulnerabilities. On the other hand, asymmetric cryptography, also known as public-key cryptography, operates with a pair of distinct yet mathematically linked keys. This method involves a public key, accessible to anyone, and a private key, tightly held by its owner. To borrow an analogy, picture a locked mailbox where anyone can drop a message using the public key, but only the owner possesses the private key to open and access the contents within. With this mechanism, data can be encrypted with the recipient's public key and decrypted with their private key. One notable advantage of asymmetric cryptography is that it eliminates the need for secure key exchange. Nevertheless, it is computationally more intensive, making it less suitable for encrypting large amounts of data compared to symmetric cryptography. Since our priority is to make the encrypt the data as securely as possible, and no large amount of data to be encrypted, a public key cryptography will be perfect for this algorithm. An algorithm is said to have a "security level of n bit" if the best known attack requires 2^n steps.[14] The table I compares the security level in bits provided by various cryptography algorithms.

By analysing the above table, it can be concluded that the RSA algorithm is perfect for the objective of this thesis. The RSA algorithm is a asymmetric cryptography algorithm which gives it an edge over the AES and the 3DES algorithms. Even though the RSA algorithm requires sizeably longer keys than the Elliptic Curve Algorithms and the same length of keys as the Discrete Logarithm Algorithms; the complexity of the algorithms is what makes the difference. The other algorithms are way to complex and require far more computational power when compared to the RSA algorithm. Such complexity is not required for this application.

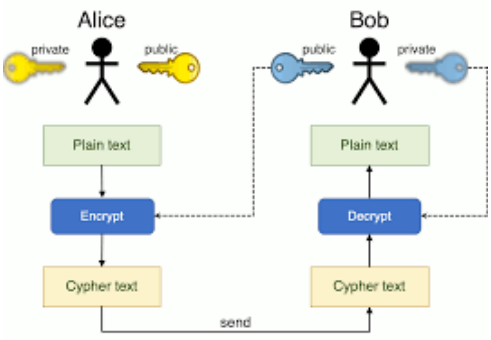


Figure 3. RSA example [2].

V. The RSA Algorithm

RSA or Rivest-Shamir-Adleman, named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman, is a widely adopted cryptographic algorithm. The RSA Algorithm is a cryptographic algorithm that encrypts and decrypts messages through the use of public and private keys. It is based on the mathematical problem of factoring large integers into prime factors, which is believed to be difficult for classical computers. In RSA, each user has two different keys - a public key and a private key. The public key is shared with others, whereas the private key is kept secret. Considering the widely used Alice and Bob example; When Alice wants to send a message to Bob, she will encrypt it using Bob's public key. Bob can then decrypt the message using his private key. This way, even if someone intercepts the message, they won't be able to read it since they don't have Bob's private key. This is demonstrated in the figure 3. Similarly, Bob can send an encrypted message to Alice using her public key. Alice can then decrypt the message using her private key. This method ensures that only Bob and Alice can read the messages they send to each other. RSA Algorithm is based on the difficulty of factorizing large numbers into two prime numbers. This makes RSA highly secure and strong against most of the attacks. It has a security strength of 2048 bits and above, which makes it practically impossible to break using conventional computing methods. However, with the advancement in technology, even 2048 bits security may not be enough to withstand attacks from quantum computers. In such cases, RSA will have to be replaced with newer and stronger cryptographic algorithms. Breaking RSA is considered to be a daunting task due to the immense computational power and time required. RSA encryption can only be broken by factoring the large composite number into two prime numbers. This means that if the key used is long enough, it may take billions of years to break the encryption.[14] Currently, it is believed that it will be possible to factor 1024-bit values within the next 10 to 15 years with the help of quantum computing. To minimize the risk of such an attack,

Table I
DIFFERENT SECURITY LEVELS FOR CRYPTOGRAPHY ALGORITHMS [14].

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer Factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete Logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic Curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric Key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

it is recommended to choose the RSA parameters of 2048-4096 bits. Even though RSA is considered to be highly secure, there have been instances where security breaches have occurred. One such attack was the Bleichenbacher's attack, which exploited a vulnerability in the RSA encryption implementation. This resulted in the attacker gaining access to secure information, which highlights the importance of proper implementation of RSA. Another challenge is the susceptibility of RSA to side-channel attacks. These attacks can occur when an attacker measures the physical characteristics of the system, such as power consumption, to gain information about the key. Therefore, securing the implementation of RSA is essential to prevent such attacks. While the RSA algorithm offers robust security, it has some challenges that make its implementation complex. One of the biggest challenges is computational complexity. Generating large prime numbers and performing complex operations can be time-consuming and resource-intensive. Additionally, key management can be challenging, especially when dealing with a large number of users. Safe key storage and distribution are necessary to prevent unauthorized access. The following steps illustrate the generation of the key for the RSA algorithm:

1) Select two distinct prime numbers; Assume they are p and q.
2) Compute their product "n" such that n = p*q.
3) Calculate the Euler's totient function $\varphi(n) = (p-1)*(q-1)$.
4) Select an "e" such that $0 < e < [\varphi(n)]$ and e & $\varphi(n)$ are coprime i.e $gcd(e, \varphi(n)) = 1$.
5) Calculate a "d" such that $d.e \equiv 1 mod \varphi(n)$
6) (n,e) is the Public key and is used for encryption while (n,d) is the Private key and is used for decryption.

Once the public and private keys are calculated, the messages can be encrypted and decrypted as follows: Let x be the data and y be the encrypted data. Then, the encryption is done as $y = x^e \mod n$ while the decryption is done as $x = y^d \mod n$.

VI. DISCUSSION AND FUTURE WORK

Even though the RSA algorithm seems the best option for securing the JTAG debug port, there are some limitations when implementing it using VHDL. As discussed in the previous section, a proper RSA implementation would require keys that are atleast 1024 bits long. But, the integers in VHDL 2008 are capped at 32 bits while the integers in VHDL 2019 are capped at 64 bits. This obviously poses an issue as the key can not be declared in VDHL. For this, there would be a need to create new VDHL libraries which support such large integers. The implementation of the RSA algorithm also involves many complex mathematical operations, e.g. it involves the exponential operator, which is cannot be synthesized by an FPGA. Hence this needs to be worked around when implementing the RSA algorithm using VHDL. Another thing that comes into one's mind while implementing such an asymmetric key algorithm is securing the keys by means of a certificate. But one can argue that this is not required with the RSA algorithm. The keys of a proper implementation of RSA algorithm are atleast 1024 bits long and factorizing the public key in order to obtain the private key is almost impossible. Hence, securing the public key with a certificate would just add unnecessary computational effort. One of the future works in order to implement this, as discussed before, is creating new libraries in VHDL. Creation of a server to pre-compute and store the keys could also be done. Although the keys for a specific implementation would be stored in the implementation itself, creating such a server could reduce the effort of computing the keys every time just before the implementation. Hence, here it has been discussed how can the encryption of the JTAG TAP Port can be done through hardware based encryption. Various methods have been discussed, compared and the importance of the RSA algorithm for this application has been discussed.

REFERENCES

[1] Accessed on October 21, 2023. URL: https://www.allaboutcircuits.com/technical-articles/jtag-test-access-port-tap-state-machine/.

[2] Accessed on October 6, 2023. URL: https://www.coengoedegebure.com/surviving-an-infosec-job-interview-cryptography/.

[3] Accessed on October 1, 2023. URL: https://www.xjtag.com/about-jtag/what-is-jtag/.

[4] Luca Baldanzi u. a. *Cryptographically Secure Pseudo-Random Number Generator IP-Core Based on SHA2 Algorithm*. Applications in Electronics Pervading Industry, Environment, Society – Sensing Systems und Pervasive Intelligence), 2020. ISBN: https://doi.org/10.3390/s20071869.

[5] Gregor Benz und Andreas Siggelkow. *Implementation of a GPS and GSM module into a Zynq Z7 SoC based emulator tracking system*. Workshop der Multiprojekt-Chip-Gruppe Baden-Württemberg, 2020.

[6] Swarup Bhunia, Sandip Ray und Susmita Sur-Kolay (Editors). *Fundamentals of IP and SoC Security: Design, Verification, and Debug*. Springer, Cham, Switzerland, 2017. ISBN: 978-3-319-50055-3.

[7] Wei Hu u. a. *An Overview of Hardware Security and Trust: Threats, Countermeasures and Design Tools*. IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS und SYSTEMS, VOL. X, NO. X, 2021.

[8] *IEEE Standard for Test Access Port and Boundary-Scan Architecture*. IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001) , vol., no., pp.1-444, 2013. ISBN: doi: 10.1109/IEEESTD.2013.6515989.

[9] Zhengato Jiang, Yang Zhan, Dan Chen und Yumin Wang. *Two methods of directly constructing probabilistic public-key encryption primitives based on third-order LFSR sequences*. https://doi.org/10.1016/j.amc.2005.01.097: Applied Mathematics und Computation, 2005.

[10] C.F. Kao und H.M. Chen. *Hardware-Software Approaches to In-Circuit Emulation for Embedded Processors*. IEEE Design und Test, 25 (5): 462 - 477, 2008.

[11] Bariş Bülent Kirlar und Melek Çİl. *On the k-th order lfsr sequence with public key cryptosystems*. https://doi.org/10.1515/ms-2016-0294: De Gruyter, 2017.

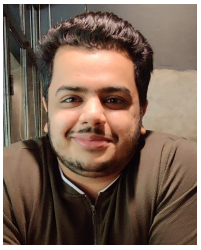
[12] Paul Kocher u. a. *Security as a new dimension in embedded system design*. IEEE Proceedings. 41st Design Automation Conference, 2004. ISBN: 1-51183-828-8.

[13] Kyungroul Lee, Yeunsu Lee, Hyeji Lee und Kangbin Yim. *A Brief Review on JTAG Security*. DOI: 10.1109/IMIS.2016.102: 10th International Conference on Innovative Mobile und Internet Services in Ubiquitous Computing (IMIS), 2016. ISBN: 978-1-5090-0984-8.

[14] Christof Paar und Jan Pelzl. *Understanding Cryptography*. Springer, 2010. ISBN: 978-3-642-04101-3.

[15] Keun-Young Park, Sang-Guun Yoo und Juho Kim. *Debug Port Protection Mechanism for Secure Embedded Devices*. DOI: 10.5573/JSTS.2012.12.2.240: Journal of Semiconductor Technology und Science, 2012.

[16] Sang Guun Yoo, KEUN-YOUNG PARK und Josphine Kim. *Software Architecture of JTAG Security System*. WSEAS Transactions on Systems, 2012.



Soham Sanjay Dekhane received his B.Tech. degree in Electronics and Telecommunication Engineering from Symbiosis International (Deemed) University, India in July 2021. Since September 2021, he is pursuing his Master's degree in Electrical Engineering and Embedded Systems at Hochschule Ravensburg-Weingarten.

Andreas Siggelkow received the academic degree Dipl. -Ing. degree in 1988 from the University of Karlsruhe. In 1996, he obtained his doctorate at the University of Stuttgart for Dr. -Ing. From 1996 to 2007 he worked for Infineon on specifications for base-band processor ASICs. Since 2007, he is a professor for ASIC-Design and Computer Architecture at the Hochschule Ravensburg-Weingarten.