SoCs in New Context - Look beyond PPA

WHITE PAPER

Pawan Kumar Fangaria
Founder & President

Fangaria Associates
1202, ATS Greens – II
A 58, Sector – 50
NOIDA – 201 307
www.fangarias.com

If we look back in the last century, performance and area were two main criteria for semiconductor chip design. All design tools and flows were concentrated towards optimizing those two aspects. As a result, density of chips started increasing and power became a critical factor. Now, Power, Performance and Area (PPA) are looked together as the prime criteria for SoCs. Since the beginning of this century the semiconductor industry (including technology, design and software) worked tremendously to optimize PPA for semiconductor chips; the latest technologies being FinFET and FD-SOI.

Today, we have started seeing temperature as a key criterion for consideration in the semiconductor design. Like PPA, temperature acts as a basic criterion at the device and chip levels. We are seeing state-of-the-art tools in the market for thermal analysis of chips and packages. Temperature is a key criterion for hand-held mobile, automotive, and storage devices and therefore for SoCs in that space.

While PPA and temperature (PPAT) definitely need to be looked at as the base criteria, my emphasis in this article is to look at the SoCs in a new context today where there are other major factors which can dominate over these basic criteria; in fact some of the other major factors effectively drive the PPAT for SoCs. There is already lot of work done for PPA and is on the table to be exploited and used in the larger context of SoCs. So, let's look at the major factors which drive modern SoC design.

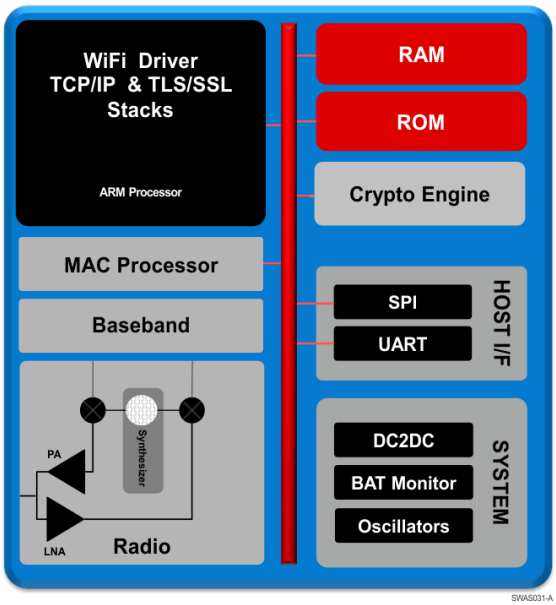
Target Segment: The days when one particular processor like **Intel** **Pentium** used to address most of the computing market needs around the world are no more. Today, we have multiple segments within one market. For example, within computing or processing space, we have desktops, laptops, tablets, smartphones, and so on. Each of these addresses a particular segment in the computing space and can have varying needs. A desktop processor can be less power efficient than a tablet or smartphone processor. Similarly, an SoC for automotive application can be less area efficient than an SoC for smartphone. Again, even for a particular segment it's not one market across the geography, the markets are further segmented across the geographical regions. A live example is about smartphone markets in USA, China, India, and so on; they are different. Hence, an SoCs should to be planned according to its target segments for what is needed in that segment, and more importantly for how long that design can survive in that market. Otherwise, you may provide the best PPA, but it can still fail in a particular market. Again, power and performance have to balance against each other; today there is no more leeway to gain on both fronts without cost and other implications.

Cost: Today, cost no longer rules the market. The market drives the cost. At the same time, wafer cost at lower nodes is increasing to an extent that the cost per transistor may not reduce substantially with further technology scaling. So, the SoCs need to be architected with appropriate functionality according to the market need and the cost which it can absorb. It's extremely important to plan the BOM (Bill of Material) upfront according to the cost and profitability. An SoC with similar functionality can have variants with different fabrics and PPAs for different markets according to their cost structures. A recent example is about **Qualcomm** launching its mid-range lines of 4G baseband processors, Snapdragon 618, 620, 415 and 425, specifically to compete in China market where **MediaTek** is aggressively gaining in 4G LTE chipset market with its low priced chipsets. Also, **Intel** is eyeing emerging markets with its low-end **SoFIA** processors. The upcoming IoT market will further establish 'cost' as a major factor for SoCs, because 'low-cost and high-volume' will be the key characteristic of various segments in the IoT market.

A more important observation related to cost, as I see it from business angle, is that the companies (or divisions in large companies) need to be swift in aligning their product line, R&D, procurement, and manufacturing processes according to the market segment they serve; otherwise they can never meet the cost structure of the market. It is okay for **Intel** to initially indulge in the so called "contra-revenue model" to gain mobile market share, but gradually, rather rapidly, it has to align the mobile product line according to the cost structure of that market.

Functionality: After firming up the top level business strategies for target segment and cost, the actual stage comes where an SoC is architected according to the requirement and driven to implementation. The functionality must come as the top consideration in implementation because that will justify the cost and target segment as explained above. One has to consider, what kind of CPU and with how many cores should be employed, is a GPU required, how much on-chip and expandable memory should be sufficient, memory controllers, interfaces, communication components, and many more. Today, there can be hundreds of functional components on an SoC and there is a lot to choose from for each component. This mandates to decide on the functionality you are going to support for a particular segment at a particular cost.

IP Integration: Once the functionality is defined, not all components can be done by one company. There comes IP for various components supplied by vendors across the world. So, here the actual exercise is to choose the best PPA optimized IP for your SoC and best integration methodology for overall PPA optimization of the SoC. This optimization is at a different level where you have to architect the data traffic and communication between different components in most optimized manner to consume lowest power and have lowest latency with minimum congestion in the network. There is Network-on-Chip (NoC) available which can be utilized to manage traffic and minimize power consumption of the overall SoC. Here is an example of how **Texas Instruments** used **Arteris FlexNoC** in its **SimpleLink Wi-Fi Family** of SoCs for internet-on-a-chip solution for IoT market in home automation, safety and security, energy harvesting, industrial M2M and wireless audio streaming. This is well architected with NoC fabric to work at extremely low power. The NoC is utilized to shut down the components which are not required for a particular mode of the chip's operation.



[Courtesy Texas Instruments: TI CC3100 Hardware Overview]

This is a very simplistic, but smart design. Imagine a design where there can be several digital and analog components and high speed interfaces that connect wires getting into and coming out of the analog IPs at different levels of voltages. The floorplanning of those IPs, IOs and busses are critical along with the software that can model the channels in the floorplan. Also, while selecting an IP, looking at its PPA in isolation is not sufficient. There can be situations where more IPs when combined together can produce innovative results. Let's keep that aside for a more detailed article later.

System Performance: It you count on a CPU performance, it's simply the product of IPC (instructions executed per clock) and the clock frequency. With the technology scaling, clock frequency has almost reached its limits; although with significant increase in leakage power at lower technology nodes. Also frequency itself has implications on power. The other avenues to increase CPU performance by increasing IPC include techniques such as ILP (instruction level parallelism). Several other techniques such as SIMD (Single Instruction Multiple Data) have been used to reduce the number of instructions for a task.

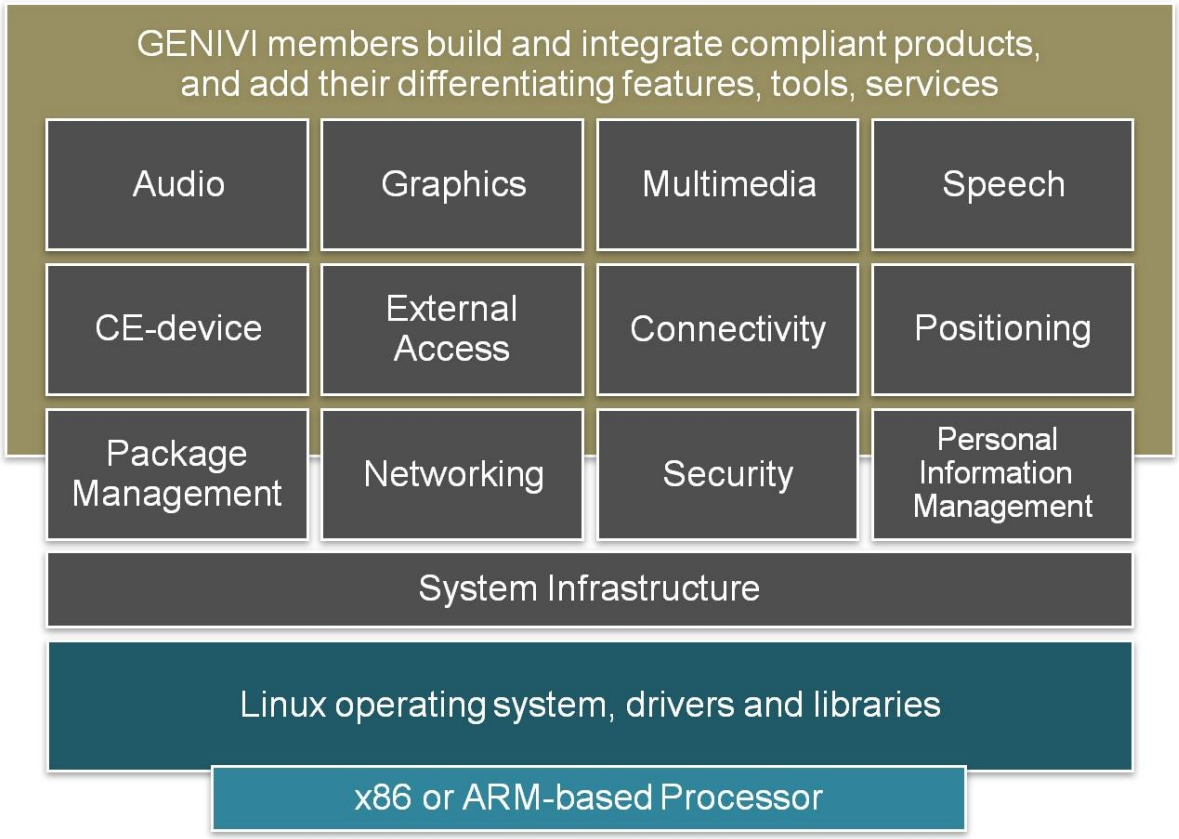
That was the case for a single CPU. Today, we need to look at the performance in terms of the whole system's performance. The CPUs can have multiple cores which provide high performance with optimized power. However, to exploit the multi-core architecture, several aspects have to be considered in the SoC design. These aspects may include the number of cores ideal for an SoC requirement, special cores for graphics, speech recognition, communication protocols, cache subsystem architecture, RAM (Random Access Memory), memory access mechanisms, and so on. So, one has to consider actual definition of performance as "time taken to execute a task" and consider the complete system performance while designing an SoC. Also, considerations have to be made for how different types of software would make use of the multiple cores.

Size: The size of a chip should not be confused with the density of transistors a technology node can provide. It's the architecture and space utilization on the chip that matters. So, it's the absolute size of the chip which needs to be considered; can it be architected with required functionality within the parameters of the size for a particular market segment. If a robust architecture can satisfy the size with the required PPA at a higher technology node, then that's the best scenario. One does not need to use lower node process unless essential. In cases of GPUs where there are parallel workloads, higher density of transistors (and hence lower technology nodes) can definitely improve performance almost linearly; so lower technology should be considered there. From a business perspective, size can be extremely important in wearable segment of mobile and IoT market. So, size of the actual SoC is an important criterion to consider before architecting.

The Target Segment, Cost, Functionality, IP Integration, System Performance, and Size as described above are the key business aspects and technical considerations that drive the architecture of an SoC for its best optimized performance within the given economic scenario. There are other criteria as described below that are essential for the SoC's quality and robustness to it to survive in today's global environment.

Hardware, Software, and Embedded Software: Today, before you architect an SoC, you have to think about how it will be driven by the software systems sitting on it and make

provisions for those. It has to account for interfaces with other multimedia, graphics, networking, and connectivity devices and software. Accordingly it has to be targeted for particular applications such as IoT, wearable, medical, automotive, and so on.



[GENIVI Software Architecture, courtesy Mentor Graphics, GENIVI]

Above is an example of a **Linux** based open source software architecture promoted by GENIVI in the automotive space. This architecture is supported on the reference boards available from Renesas, Texas Instruments, and Freescale.

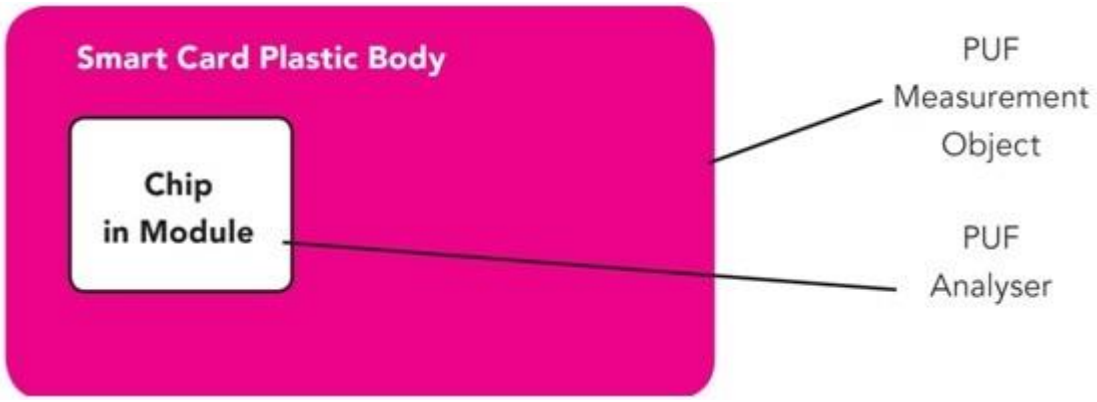
The embedded software again is an essential part of any software for SoC. It is heavily dependent on the underlying hardware and has to be designed according to the application. As we move up from the hardware up to application level, the embedded software transforms the data through Hardware Abstraction Layer (HAL) and its corresponding APIs, OS and its corresponding APIs, Communication middleware and its corresponding APIs, and the application software itself. The embedded software needs to be optimized by exploiting the characteristics of the underlying hardware and re-used in several similar systems as far as possible.

The idea is to architect a complete system including hardware, software and embedded software; i.e. a complete ecosystem. One who can do that and supply a complete SoC with authority can be the winner. Why **Apple** is the most valuable company today? One may criticize it for not being open, but the truth is that Apple is good at both hardware and software. **Google** and **Microsoft** are good at software, but not hardware. Google does great innovation in hardware, but has not been able to pursue it to practice; it has not been able to encash on Google Glass, for example. **Samsung** is good at hardware, but not at software; they have realized that they need to build their own mobile OS, but have long way to go.

An SoC is not a hardware chip anymore; both software and hardware are its core components. Hence to develop an effective SoC, one has to have core competence in both software and hardware.

Connectivity: In today's highly connected world, operating at frequencies ranging from baseband to RF in different segments, the right connectivity solution to work with the SoC has to be investigated and planned in the beginning during the architecture stage. Moreover, there are various protocols such as **Wi-Fi**, **Bluetooth**, **ZigBee**, and so on in the wireless domain, and several M2M protocols in the IoT space. Provisions for the right band and the right kind of protocols to be supported have to be planned during the SoC architecture stage. A versatile SoC with programmable microcontroller, appropriate memory and connectors' support, and support of complete frequency range applicable for a particular segment can become far more successful than an SoC connecting with only some portion of the intended devices in that segment.

Security: In an IoT world, even a small chip in a device located in a remote corner of the world can be globally connected through M2M connections and connection with the internet. In this multi-point connection, every point-to-point connection can be vulnerable to cyber attack. The security aspect has to be dealt with at hardware as well as software level for the whole of the interconnection infrastructure. The data has to be protected at the device level and its authenticity maintained during its transfer. At the device level the data has to be restricted for access only to the authorized individuals; and that has to be done at the source in the hardware, i.e. your SoC. Moreover, an SoC or an IP also has to be protected from being reverse engineered, duplicated or counterfeited. There are several methods which can be employed to secure SoCs. For example, **PUFs** (Physically Unclonable Functions) can be instantiated on the chip for device authentication and providing secure keys. PUFs can be successfully used to protect and secure memories, smartcards, USBs, and other mobile devices.



[PUF: Protecting Smartcard ICs. Source NXP]

NXP has successfully used PUFs to protect next-generation smartcard ICs. During production or personalization, the IC measures its PUF environment and stores this unique measurement. From then on the IC can repeat the measurement as and when required to check if the environment has changed, thus protecting the card.

Side Channel Attack (**SCA**), which is non-invasive, is a new kind of threat for ICs. In this, information can be obtained out of a chip based on its power profile, electromagnetic

analysis, or even timing analysis. Such attacks have to be prevented by deploying security at the physical level, may be at the leaf cells of the design. As an example, by preventing any variation in power to be detected during the circuit operation, one can secure the chip from SCA on power variation during switching of the circuit. There are newer methods evolving to address security at the SoC level; several terms are in use today such as **TRNG** (True Random Number Generators), **Root-of-Trust**, **watermarking**, and so on. Several methods are being used to detect hardware **Trojans**.

The point is that for a particular SOC to be designed for certain applications or environments, its security aspects must be thought of during the architecture stage and the same must be implemented from the base level.

Reliability: In SoCs at ultra-low process nodes (20nm and below) having transistors at extremely low noise margins, and SoCs taking care of several functions, reliability cannot be denied or considered after functional implementation. In the pursuit of PPA optimization, reliability is often overlooked. High performance, at high frequency will consume high power and can become a source of heat leading to electro migration and other complications. If the heat is not estimated and rated, and provisions made for its diffusion, then it can deteriorate the life of the device sooner than later. There are tools available to estimate power dissipation, noise, and reliability. Although a CPU's real power dissipation can be computed, actual consideration should be **TDP** (Thermal Design Power) for determining temperature ranges and designing appropriate cooling systems where needed. Recently, **Intel** designed its **CoreM** processor at 14nm technology node that has a TDP range between 3.5W (with down freq 600 MHz) and 6W (with up freq 1.4 GHz) with a nominal value of 4.5W. Interestingly, **Apple** used Intel's CoreM for its new MacBook and didn't need to put a fan in it.

Verifiability and Testability: Various **DFT** methodologies such as scan chain, built-in self test (**BIST**), **MBIST**, and so on are well known for making a design testable. And that has become a usual practice. With an increase in design complexity, now a day there is emphasis on making a design verifiable. The focus is on the micro-architecture that should simplify the verification process. This includes synchronization of the design, minimization of cycle based logic, ensuring the design to be **CDC** (Clock Domain Crossing) safe, avoiding complex interfaces, and so on. The Verification IP (VIP) should be easily migrated between different design levels, e.g. RTL to gate level. A better verifiable design should also be easily debug-able. For this interfaces need to be defined formally and cleanly. Assertions need to be added at various places to verify conditions.

Serviceability: In large SoC different types of errors may occur at any point of time due to various reasons; however that should not initiate replacement of the whole SoC. There must be provisions in the SoC for easy diagnosis of the problems and their correction. The components that are likely to fail should be easily identifiable and reparable or replaceable.

There can be soft errors as well as hard errors in an SoC. Memories such as DRAMs are very susceptible to errors due to large data storage and activities at extremely high rate of data transmission. These errors can be soft errors, hard errors or retention errors.

Typically, **ECC** (Error Correcting Code) circuitry is used in the SoC to correct the errors that could be caused in the data. The ECC can be used in various modes of operations depending on the location or errors. For example, ECC scrubbing can be used to check the whole

memory array and correct all single-bit errors. Memory sparing is another activity which can instantly replace any failing memory by spare memory in the system. Similarly there are other techniques such as data/address parity, cyclical redundancy checks (**CRC**) etc. which can be used in the SoCs to address problems and maintain data integrity. Power-on-Self-Test (**POST**) and Built-in-Self-Test (**BIST**) are the techniques which can detect hard errors. The hard errors cannot be corrected; they have to be mapped out of the usable area by the operating system.

The idea is to make provisions in the SoCs to have appropriate methods and circuitry for different types of components to be diagnosed and corrected as and when required.

This whitepaper provides a general preview of several factors which are important to consider in today's SoCs, unlike only power, performance and area in earlier chips. Each factor in itself is a complete field in semiconductor and there are several companies providing solutions for the same. It's not possible for a single company to have solutions for all of these. It requires a collaborative approach for today's SoCs to find the best possible solution from different companies specializing in these areas and then architect them in best possible manner.