Applying Heterogeneous Concepts in Embedded / IoT Systems

Ricardo Castanon Ureno

***Abstract*—Technology has quickly progressed with the aid of the continuation of Moore's law, which states a doubling of transistor density every two years. However, the complementing principle of Moore's Law, the Dennard Scaling, which states that power density would stay constant as transistor sizes shrunk, stopped being true in 2004. With the breaking down of Dennard Scaling, new challenges were introduced including in power management and heat dissipation, particularly for smaller devices like embedded systems. This ushered in a search for innovative solutions, eventually leading to heterogeneous design, where various processing units like CPUs, GPUs, and FPGAs are integrated into a single device. With the introduction and advancement of heterogeneous systems, new capabilities are becoming possible within small, but fast and energy-efficient devices. However, with the high complexity of the systems and with the power and heat-dissipating constraints, multiple challenges are involved. It is vital to help advance heterogeneous embedded systems since it is such versatile technology; advancing heterogeneous systems can help propel various sectors.**

Keywords-component; heterogeneity; embedded systems; inernet of things (IoT); power constraints; heat dissipation

I. INTRODUCTION

The introduction of heterogeneity addresses several critical challenges that were not properly addressed by homogeneous multicore solutions or "dark silicon" practices [1]. Homogeneous solutions became wasteful by all cores being available for usage but not being fully utilized. Homogeneous systems are limited due to thermal constraints which require thermal throttling [1]. Thermal throttling is the process of lowering the component's clock speed/frequency to slow down processing. However, this would cause significantly worsened performance [1]. As technology has advanced to involve multiple high computing specialties, such as artificial intelligence, graphics rendering, and real-time data processing, it has become important to be as efficient as possible [2].

Heterogeneity was introduced after the realization that a one-size-fits-all approach to processing elements is no longer sufficient to meet more specific functionalities. Thus, heterogeneous systems and their underlying concepts have emerged at the forefront of advanced computing. At their core, heterogeneous systems are unified in a single device and include multiple elements such as CPUs, GPUs, FPGAs, and specialized accelerators [3]. Each element is used in collaboration with each other, taking advantage of the unique strengths available in the overall computational workload. Heterogeneity acknowledges that certain tasks are best suited for specialized processors, giving rise to a more efficient device.

This change in system architecture does not come without challenges. Heterogeneous systems are highly complex in both the hardware architecture and software programmability aspects. The multidisciplinary nature of heterogeneous system development, involving engineers with diverse expertise, creates synchronization challenges between the several components in the overall system [3]. Such issues can lead to project management difficulties and cause problems when time constraints are important [3]. Heat dissipation and power consumption are also notable issues, specifically for small devices such as embedded systems. These are factors that are important to consider as embedded systems are small in nature, are often found in unsuitable environments, often have small batteries, and rarely have internal cooling.

In response to these challenges, a spectrum of solutions is being explored and implemented. Such solutions include creating new software development strategies, creating new programming paradigms, improving performance prediction, and strategically choosing which processors/accelerators to use depending on each task.

II. HETEROGENEOUS CONCEPTS

Heterogeneous systems can be composed of multiple elements, each having their own unique strengths and specific use cases. The goal of using such a wide range of components on a single device is to use the available resources as efficiently as possible to achieve the best possible performance.

There are two main types of heterogeneity, functional heterogeneity, and performance heterogeneity [1].

Functional heterogeneity occurs when different kinds of processing units with different instruction architectures are used within the same device [1]. Example processing units include:

- Central processing units (CPUs)
- Graphics processing units (GPUs)
- Tensor processing units (TPUs)
- Neural processing units (NPUs)
- Field programmable field arrays (FPGAs)

- Digital signal processors (DSPs)
- Specialized accelerators

All units work in collaboration with one another as part of the same system but how exactly they are implemented depends on the specific use case.

Performance heterogeneity occurs when cores have the same instruction set architecture but have different underlying micro-architectures [1]. Sharing the same instruction set architecture but having differing micro-architectures gives added flexibility in the areas of execution efficiency, power consumption, and overall processing capabilities. Having different micro-architectures allows systems to utilize the specific areas in the system that best fit what it is being used for. For example, one micro-architecture may offer high throughput which would be beneficial for cases that require rapid data processing. Another micro-architecture may instead be better suited to perform parallel tasks at the same time. These are just two examples of contrasting micro-architectures that could be present in a performance heterogeneous system. However, many more micro-architectures that bring their own benefits and use cases can be implemented in the same system.

III. Embedded and IoT Systems

Heterogeneity can be applied within a wide range of computing systems as it can bring improved performance in several types of applications. Two notable areas in which heterogeneity can specifically be applied are embedded systems and Internet of Things (IoT) systems.

A. Embedded Systems

Embedded systems are computer systems that combine computer hardware and software to be utilized for a specific function. They are often found within larger, more complex systems. They are low-cost, low-power consuming, small computers that are embedded within other systems [4]. Embedded systems are typically composed of processors, power supply, memory, memory ports, and communication ports [4]. They transmit data between other system peripherals, including other embedded systems, that make up the macro system. Embedded system functionality can vary a lot depending on how it is being used and implemented.

Embedded systems are currently being applied in several sectors for several uses. The ways that they are being used are continuously growing as technology is advancing. Some of the current use cases include [4]:

- Automobiles: Modern cars are composed using several computers, in the form of embedded systems. Each embedded system has its own functionality but is vital for the proper performance of the car.
- Mobile phones: Embedded systems are often used to implement Graphical User Interface (GUI) software and hardware, Operating Systems, and Input/Output (I/O modules.
- Industrial machines: Sensors within embedded systems, or in the form of embedded systems, are commonly found in industrial machines. These sensors may be used for monitoring the system or to apply automation.

Due to the extensive architectural and software organization ways possible for embedded systems, applying heterogeneity effectively in their architecture can produce more efficient systems.

B. Internet of Things

Internet of Things (IoT) is a network of interconnected devices in which the devices can communicate with each other and the cloud [5]. IoT devices are typically embedded with other technological components such as sensors and software. IoT networks consist of web-enabled smart devices that use embedded systems to manipulate data within the network [5]. A main benefit that IoT provides is that IoT devices can do a lot of tasks in the background, without human operators having to intervene throughout the process. This saves a lot of time and ultimately helps improve the lives of humans.

Similarly, to embedded systems, IoT technology is a rapidly growing technological sector and has a lot of use cases. Some of the current use cases include [5]:

- Automobiles: Modern cars are composed using several embedded systems, many of which are interconnected through Internet of Things. IoT is being used more in the automobile industry as the autonomous vehicle sector grows.
- Wearable smart technology: Technology such as smartwatches can be used as an IoT device for multiple reasons including for health applications.
- Household appliances: Adding intelligence to interconnected household appliances such as vacuums, laundry machines, and refrigerators makes these devices easier to use. This makes humans' lives easier by simplifying tedious household tasks.

IoT networks consisting of a wide range of devices working with one another are an example of heterogeneity. It is critical to create heterogeneous IoT devices to be as efficient as possible to attain the best performance. Likewise, communication between the different devices must also be effectively arranged. The unique heterogeneity in IoT makes it difficult to proficiently execute but doing so adds to the benefits generated by IoT.

Embedded system and Internet of Things system usage is ever-expanding and brings many benefits to their use cases and human lives. As automation expands and data processing moves toward the edge, making these systems as effective as possible will unlock new possibilities. Applying heterogeneity to these systems will allow the systems to become more intelligent, environmentally friendly, smaller, and more power efficient [3].

IV. Application of Heterogenous Concepts in Embedded / IoT Systems

Tremendous progress has been made in heterogeneous computing which has been applied to embedded and IoT systems. Heterogeneity allows cramming more components together into smaller systems, which results in smaller yet more

efficient systems. Achieving this sounds great and is a possibility with the continuous shrinking of transistors and improved silicon manufacturing processes. This allows for improved systems while maintaining cheap costs. However, producing these proposed heterogeneous systems is easier said than done. Implementing heterogeneity in these systems brings about several unique challenges. Exploring these issues as well as some proposed solutions gives a better idea of the current state of applying heterogeneous in embedded and IoT systems.

A. Software Complexity

Heterogeneous hardware architecture is highly complex as many micro-architectures and devices are interconnected in the same system. With this hardware complexity, also comes software complexity which is more difficult to address [3]. Combining the diversity between the hardware and software architectures creates problems corresponding to time-to-market, cost decrease, transparency, scalability, flexibility, and maintainability [6]. Designing parallel software for homogeneous computing systems was already a complicated enough challenge. Software development for heterogeneous computing takes it to another level. Notable questions that arise when producing heterogeneous systems are [1]:

- Which core or accelerator is best to use for the application?
- How is an application executed using multiple cores and accelerators?
- How can the system remain within the Thermal Design Power (TDP) constraint?
- How can battery life be maximized?

These questions are not simple to theoretically answer, executing the solutions is an added level of complexity.

Additionally adding complexity to heterogeneous systems is the embedded system-specific operating system [3]. A popular embedded system operating system is the GNU/Linux system. This system is complex and is nothing like utilizing an operating system for another device such as using Raspbian for a Raspberry Pi [3]. Using embedded Linux requires everything to be done within the command line. Very little is instantly available in the embedded Linux OS, requiring everything to be built from the ground up [3]. Some external tools and manufactured prebuilt boot images have been made available to automate embedded Linux system setup [3]. However, it is still certain that the build will need to be modified to meet the specific needs.

B. Project Management

Applying heterogeneity to embedded systems and IoT outputs many benefits including producing smaller devices. However, what does not shrink is the level of expertise needed for teams producing such complex systems.

Heterogeneous systems are highly complex and require highly experienced people from multiple areas. Creating a heterogeneous system requires performing good engineering in all areas [3]. This is not an easy task and is not easy for the project manager to handle. When developing heterogeneous systems, a group of engineers of varying specialties work together to implement their individual pieces together. The most strenuous task, however, is getting the whole system to work correctly [3]. To do so, extensive test and integration engineering is required. This is a labor-intensive but essential task in development [3]. These engineers must exit their comfort zones and delve into other engineering disciplines in which they are less experienced.

Due to the system being composed of several devices which several engineers specialize in, keeping barriers between each other's work and components stifles productivity. On the other hand, cutting down limits and barriers between the different areas increases productivity [3]. This can be done by increasing visibility into the internals of the different device sections. Doing so makes it easier to test and log between the different devices with varying input/output ports and varying flag states/events [3]. This will ultimately make it easier for non-specialists to understand when something goes wrong, and to assign the appropriate developer to resolve the issue.

Something that aids in designing with a smaller team is tools given by semiconductor device manufacturers. These tools can help automate the development tasks. However, this brings another factor to keep in mind, as it is important to decide just how much of the development will be done autonomously. The more a tool automates a task, the less it is understood of what is truly going on [3]. By default, these tools do more than what is required, which can cause problems in the development later down the line [3]. These autonomous developing tools can be used as a massive productivity help but leaving it all to them is not a great idea. A balance between manual implementation and autonomous implementation is the way to go.

Developing such a complex system as an embedded heterogeneous system requires a lot of planning, management, and expertise. Many factors such as the embedded OS, autonomous development, team size, and area transparency must be decided before development commences [3]. General tips for good practices include creating a functioning core platform and avoiding unnecessary features [3]. Because the development team varies in specialties and expertise, ideas must be limited, and rather have the project led by a dedicated technical project leader [3]. This will steer the project in a single, correct direction.

C. Power Consumption and Heat Dissipation

Power consumption and heat dissipation are factors that must be considered when it comes to creating heterogeneous systems. Although heterogeneous systems perform better in terms of power consumption and heat dissipation than heterogeneous systems or single-core systems, it is still difficult to stay within the low power and heat constraints needed from small-sized embedded systems and IoT systems.

Heterogeneous systems achieve better power efficiency than traditional systems because they allow a more specialized system execution depending on what the application needs in

real-time. Instead of using the same cores with the same code, different cores are used which specialize in specific functionalities [1]. The most appropriate core is used depending on what is needed. This results in very energy-efficient and high-performance execution. [7].

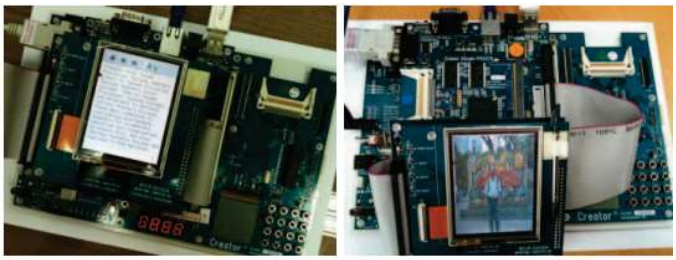
Power and heat must be monitored and limited to not go above the thermal design power (TDP) constraint. It is important to not exceed the power budget or thermal limits of a device to avoid hardware damage, inadequate performance, or even full shutdown. [1]

A solution to limiting heat dissipation is thermal throttling. This is the process of lowering a component's clock speed/frequency to meet thermal constraints [1]. However, this has its downfalls, as implementing thermal throttling significantly impacts co-execution performance. [1] A system decides when to trigger thermal throttling depending on the importance of maintaining a balance between performance and temperature control [1]. However, based on its drawbacks, it is better to limit thermal throttling as much as possible and instead choose the best application execution method using heterogeneity.

V. Research Results

A. Solving Software Complexity

Software complexity is a daunting hurdle to overcome during heterogeneous system design, but it is possible to do. Yang-Hsin Fan, et al. proposed and tested a custom-made software synthesis middleware to respond to the issues caused by software complexity [6] This middleware's functionality included automatically generating system software. Some benefits of adopting middleware on embedded systems include that it adds convenience by adding a unified interface, adding reconfigurability, adding scalability, and adding transplantability [6]. The authors created and tested their middleware schemes intending to demonstrate the feasibility of their software synthesis middleware. They conducted their tests using two systems. First, an e-book system (Figure 1a) that allows reading from a graphical user interface. Second, a digital photo frame (Figure 1b) that scrolls through pictures. Both applications were implemented using a software synthesis middleware, embedded Linux OS, 2.6.15 kernel, EXT3 file system, and miniGUI. Software synthesis of the middleware was used to automatically generate software for the systems. Both studied systems were created and demonstrated correct functionality which demonstrates using the proposed software synthesis middleware was a success. With these results, it is demonstrated that the software synthesis process can solve issues concerning the high software complexity for heterogeneous systems.



(a) e-book system (b) Digital photo frame system

Figure 1. Heterogeneous embedded system platform [6]

B. Demonstrating Power Consumption, Heat Dissipation, and Performance Improvement

Tulika Mitra tested heterogeneity's effect on runtime, speedup, and heat dissipation. Mitra used OPTIC which is a tool used to select which processors are chosen to execute tasks while minimizing thermal throttling and maximizing performance. The results were positive, showing an improvement in runtime, speedup, and heat dissipation.

First, the runtime and runtime improvements were measured by running several computing tasks. The results are demonstrated in Figure 2. The x-axis represents different computing tasks or benchmarks. The left y-axis measures the runtime for each computing task in seconds. The right y-axis measures the runtime improvement in percentage. The blue bar represents the runtime for the computing task when executed on the CPU alone. The orange bar represents the runtime for the computing task when executed on the GPU alone. The green bar represents the combined runtime when both the CPU and GPU are combined. The gray bar represents the improvement in runtime achieved when using a combination of CPU and GPU compared to running on either the CPU or GPU alone.

The average improvement for the runtimes was 19%. This proves runtime is improved when using heterogeneous systems. This is a positive sign because it shows that the heterogeneous system increases runtime in the combination of both the CPU and GPU. This demonstrates both processors are utilized more when combined. This leads to the workload being distributed more evenly between the two instead of focusing all the workload on a single processor. This leads to better performance, less heat dissipation, and decreases the chances of a processor being damaged.

A result in the graph that can be confusing to understand at first is why the GPU Runtime for CORRELATION and COVARIANCE was much much higher than for other tasks. This result is caused by what the tasks of CORRELATION and COVARIANCE consist of. Both computing tasks involve multiple high-level mathematical operations. These operations can be parallized which is a task that GPUs do a great job in. This is why the GPU runtime is much higher for these tasks compared to the others. This further demonstrates that by using a heterogeneous system, the correct processor is chosen for execution when it is more beneficial. In this case, the GPU was utilized to take advantage of its strengths in parallelization. Similarly, within a heterogeneous system, the correct processor

can be used depending on when each of its strengths can be utilized during runtime.

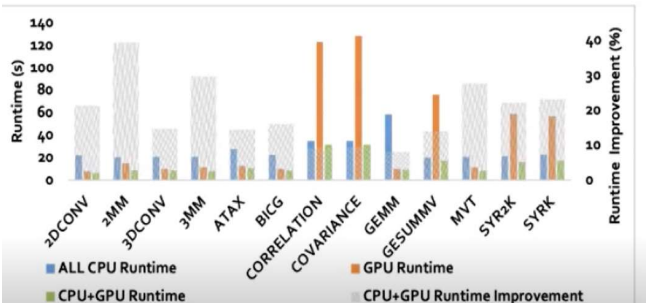


Figure 2. Comparing processor runtime when running different tasks [1]

Next, speedups between different heterogeneous configurations and a single CPU configuration were compared. This is shown in Figure 3. The first configuration is a combination of a CPU and NEON, demonstrated by the orange bar. The second configuration is a combination of a CPU and an FPGA, demonstrated by the purple bar. The third configuration is a combination of a CPU with a more complex heterogeneous system, demonstrated by the blue bar. All three configurations are compared to a CPU-only configuration. They were tested when processing 6 datasets with different neural network architectures. These different datasets are labeled in the x-axis. The y-axis indicates how much faster each configuration is compared to running the task on the CPU alone.

The graph shows positive results for all the heterogeneous configurations. Each heterogeneous configuration demonstrated a speedup improvement compared to the CPU-only configuration. The CPU+heterogeneous system configuration showed the best speedup results. For this configuration, the lowest speedup was 4.47, the highest speedup was 9.44, and the average speedup was 7.54. This proves heterogeneous systems are the best system configuration to improve speedup performance. Performance was improved regardless of how complex the heterogeneous system was.

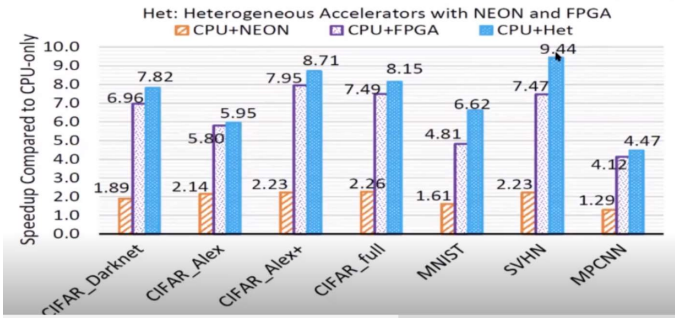


Figure 3. Comparing speedup for different architecture configurations [1]

Lastly, the CPU and GPU utilization were measured when being used inside the same heterogeneous system. As demonstrated within the other tests, splitting utilization improves performance and speedup. Paired together, Figure 4 and Figure 5, demonstrate how alternating processor utilization affects heat dissipation for the alternating processors. Figure 4 demonstrates the utilization of the GPU and CPU as time passes by and the application requires different processing types. Figure 5 demonstrates the heat map of the heterogeneous system as the processor utilization alternates. Figure 5a demonstrates when the CPU is being mainly used. Figure 5b demonstrates when the GPU is being utilized more. From Figure 5, it is easily noticeable that alternating processor utilization avoids putting most of the heat dissipation on a single processor. This can help avoid hindering system performance and help avoid system damage.

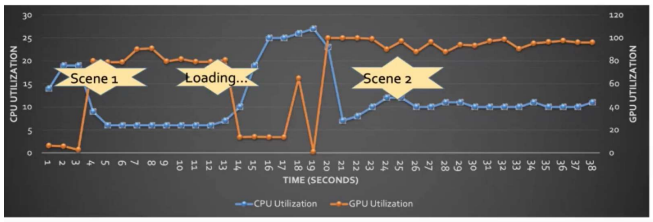


Figure 4. Comparing CPU and GPU utilization [1]

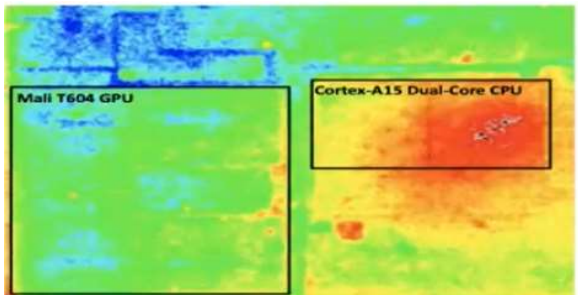


Figure 5a. Heat map of the heterogeneous system, mainly CPU utilization [1]

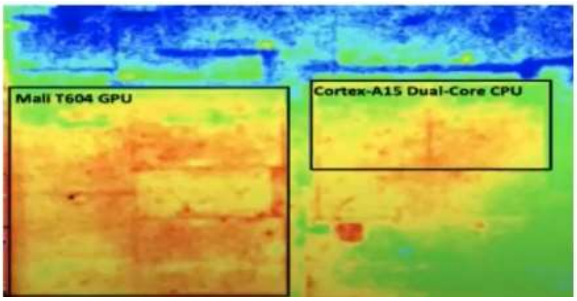


Figure 5b. Heat map of the heterogeneous system, CPU & GPU utilization [1]

VI. CONCLUSION

This study delved into the realm of heterogeneous systems, focusing on their application in embedded systems and Internet of Things (IoT) devices. Challenges posed by power constraints and heat dissipation were discussed, demonstrating why innovative solutions in the form of heterogeneous systems are important to understand and effectively produce. Lessons learned encompassed a deep understanding of embedded systems, IoT, and the significance of heterogeneity. The study unveiled the benefits of heterogeneity, especially in the context

of small computers like embedded systems but also demonstrated that the same strategies can be implemented in larger systems such as servers and neural network applications [8].

The insights gained revealed the complexity involved in implementing heterogeneity, both in terms of hardware architecture and software programmability. A notable lesson was the intricate relationship between different processing units, requiring synchronization and collaboration. The application of heterogeneity in embedded systems and IoT showcased its potential to enhance efficiency and performance in various sectors, including, but not limited to, automobiles, mobile phones, industrial machines, and wearable smart technology.

However, there are still open issues that must be addressed. A large open issue is utilization predictability to correctly predict the best execution method. Efficiently and rapidly predicting execution time allows manufacturers to get the best performance from heterogeneity. The rapid evolution of technology, with new processors, CPUs, GPUs, and NPUs emerging regularly, presents another ongoing challenge. It is difficult to create a complex system before new technology is released which then causes the produced heterogeneous system to quickly lose value. Lastly, another open issue concerning heterogeneous systems is the wide range of component manufacturers that produce the components found inside heterogeneous systems. With different manufacturers, come different manufacturing habits and different architectural methods. This can cause further issues in developing complex systems as the system components may not be easily compatible with one another.

In conclusion, while heterogeneous systems hold immense potential, their successful implementation requires addressing ongoing challenges, considering important production factors, and using the best components depending on the use case.

REFERENCES

[1] Tulika Mitra, Vishnu Pendyala. Embedded Heterogeneous Computing: A Software Perspective. (October 26, 2021). Accessed: Nov 19, 2023. [Online Video]. Available: https://ieeetv.ieee.org/video/embedded-heterogeneous-computing-SW-perspective.

[2] E. Worthman, “Trends and Opportunities in Embedded Processors and Systems.” ElectronicDesign.com. https://www.electronicdesign.com/technologies/embedded/whitepaper/21267561/texasinstruments-trends-and-opportunities-in-embedded-processors-and-systems (accessed Nov. 25, 2023).

[3] C. Funnell, “The Rise of Heterogeneous Systems and the People Problem.” eandt.org. https://eandt.theiet.org/2020/06/18/rise-heterogeneous-devices-and-people-problem (accessed Nov. 25, 2023).

[4] B. Lutkevich, “embedded system.” techtarget.com. https://www.techtarget.com/iotagenda/definition/embedded-system (accessed Dec. 9, 2023).

[5] A. Gillis, “internet of things (IoT).” techtarget.com. https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT (accessed Dec. 9, 2023).

[6] Y. -H. Fan, J. -O. Wu and S. -F. Wang, "Software synthesis of middleware for heterogeneous embedded systems," 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), Yichang, China, 2012, pp. 2084-2087, doi: 10.1109/CECNet.2012.6201427. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6201427&isnumber=6201368.

[7] A. Majumdar, S. Cadambi and S. T. Chakradhar, "An Energy-Efficient Heterogeneous System for Embedded Learning and Classification," in IEEE Embedded Systems Letters, vol. 3, no. 1, pp. 42- 45, March 2011, doi: 10.1109/LES.2010.2100802. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5672393&isnumber=5735607.

[8] X. Lin, X. Zhou, R. Liu and X. Gao, "Compare with the Traditional Heterogeneous Solution: Accelerate Neural Network Algorithm through Heterogeneous Integrated CPU+NPU Chip on Server," 2023 IEEE 3rd International Conference on Computer Communication and Artificial Intelligence (CCAI), Taiyuan, China, 2023, pp. 45-49, doi: 10.1109/CCAI57533.2023.10201248. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10201248&isnumber=10201240.

[9] Wenheng Liu, W. J. Kostis and V. K. Prasanna, "Communication issues in heterogeneous embedded systems," Proceedings of the 4th International Workshop on Parallel and Distributed Real-Time Systems, Honolulu, HI, USA, 1996, pp. 180-183, doi: 10.1109/WPDRTS.1996.557675. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=557675&isnumber=12101.

[10] S. Mostert, "Constructing a heterogeneous real-time system," Proceedings of 11th IEEE Workshop on Real-Time Operating Systems and Software, Seattle, WA, USA, 1994, pp. 34-38, doi: 10.1109/RTOSS.1994.292565. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=292565&isnumber=7232.