Applications of Asynchronous Components in Processor Design

Evan Smith
School of Engineering and Computer Science
Syracuse University
Syracuse, NY
esmith15@syr.edu

***Abstract*—The use of asynchronous components and systems within processor designs has been covered in more depth as the advantages of such approaches become clear. In this paper, we cover a survey of research into various levels of asynchronous hardware integration into processors. We also explore certain uses of such processors, including neurosynaptic applications, and propose a use case related to virtual reality.**

Keywords—asynchronous processors, event-driven sampling, IoT, neurosynaptic chips, power optimization

I. Introduction

The design of processor hardware has been, and will continue to be, in a constant state of improvement. This improvement has in the past focused primarily on increasing performance, storage, and other elements of the system by improving synchronous components such that the clock period can be decreased further and further, enabling faster computations and data manipulation. However, there is another paradigm of hardware design that is intriguing for many use cases: asynchronous design. This approach frees components from a regimented work cycle, integrates more cleanly into analogue applications, and greatly reduces power consumption.

In this paper we survey some general improvements in asynchronous processor design and then explore some of the specific applications in which asynchronous design provides advantages over traditional synchronous design. Some of these applications include communication meshes, IoT and self-powered devices, and reduced local electromagnetic environment.

II. Design Concepts

A. General Overview of Concepts

Asynchronous design is fundamentally about designing hardware in blocks. Much like object-oriented programming, this design philosophy seeks to isolate discrete components that should have a specific task and allow them to perform those tasks without intervention from an overarching global control structure. That is not to say, of course, that there is not a global design or system of monitoring and controlling the processor – that would render the hardware unusable. Instead, it passes the responsibility of requesting and sending workloads down to the work-performing components themselves.

This system of self-regulation means that components can operate at different frequencies of activity, allowing for a processor that can have several instructions or other behaviors working in parallel, much like synchronous pipelining, but without a central controller monitoring and steering the behavior of the whole system. The key change to the overall design of the processor is the removal of a global clock. This allows the designers to remove a huge number of connections that require powering at very rapid speeds over the entire area of the chip, which lowers both the footprint and power consumption.

Taking the place of the global clock is some form of handshaking protocol, which can vary based on the exact design of the hardware. The handshaking concept is analogous to a client-server relationship in systems design: the components are linked sequentially and use a series of confirmation messages to confirm that both the sender and receiver are available to perform their tasks, and then confirm that the tasks were performed properly.

This means that custom-designed components can be linked together sequentially or in networks and can communicate and perform large-scale, coordinated procedures with no global controller.

B. Initial Research - AMULET

Among the first successful implementations of an asynchronous processor was the AMULET system, introduced in 1994 [5]. Followed by several later improvements, this chip was eventually shown to be competitive with contemporary systems such as the ARM9 in processing speed and memory performance, but with significant advantages in its simple power management and reduction in electromagnetic interference.

The authors acknowledge in their paper that the applications for such a technology would be niche, but important. While a specially designed chip can approach the benchmarks of a conventional synchronous processor, the intricate web of handshaking is both difficult to design and tends to be quite application specific. This makes it hard to use in a conventional processor for consumer and commercial application, where the instruction set needs to support a large number of behaviors and peripherals, each of which would potentially need a custom on-chip implementation.

A way that the AMULET designers got past this interfacing issue is by dividing the chip into an asynchronous core that works on certain inputs alongside a synchronous controller which handled the specific telecommunications application that the chip was intended for. This pattern of dividing the processor into a synchronous and asynchronous portion was later used for other successful commercial applications that we will explore.

C. Mimicing Synchronous Behaviour

As research grew in the asynchronous space, there was a push to try and mold the technology into a more serious competitor with synchronous systems. One such step was to try and reduce the development overhead costs of designing the systems, which had limited design tools available, and are in general difficult to simulate within synchronous machines.

One attempt to make design a bit more appealing was to use existing synchronous FPGA technology and tools to design asynchronous systems [1]. This proof of concept showed that it was possible to write silicon-level compilers that take in representative code and produce VHDL for an asynchronous implementation of that code which can run directly on an FPGA.

While this is not an optimal hardware setup to demonstrate many of the benefits of asynchronous design, this work allowed for more simple iterative design and still provided savings in power consumption.

Another approach taken in research was to implement known high-level architectures using asynchronous components and designs. One such example recreated a MIPS processor asynchronously using Concurrent Sequential Processes [3]. The resulting instruction set, known as Synthesizable Asynchronous MIPS (SAMIPS), was a 32-bit processor that used the paradigm of Globally Asynchronous, Locally Synchronous (GALS) to bring a more hybrid approach to the table. The idea is to have more conventionally designed components like registers, decoders, and others be locally clocked for internal operations, but to use handshaking protocols to communicate between the components. This allows the local clocks to only run (and pull power) when the component is active with work.

Additionally, the SAMIPS processor demonstrated a way to deal with the data hazards that still arise when taking a GALS approach to design. Data hazards were avoided by maintaining a register stack containing all addresses that were actively "checked out" by another component. Control hazards were creatively handled by inventing a "coloring" of the processor. This entailed a chip-wide branch prediction that would be one color, and all incoming decoded instructions would have that color passed along with them. If at any point the branch prediction failed, the processor would change colors and notify all components of this change. While incoming instructions were still correctly colored, components would reject any residual incorrectly colored instructions if they were seen. This allowed the processor to clean out the pipeline of inaccurate or unnecessary calculations.

III. Sensor Network Applications

As has been mentioned, a key benefit of asynchronous systems is their superior power efficiency and reduction in electromagnetic emissions (EME) by cutting out the web of pulsing clock signal lines found in synchronous systems. This combination of low power operation and lower likelihood of interference with delicate hardware made it a great option for managing and processing sensor networks.

A. SNAP/LE

The low energy sensor network asynchronous processor (SNAP/LE) was developed with the goal of producing a processor with a very power-efficient idle state and a rapid wakeup [2]. Combined, these attributes are ideal for distributed, small-scale sensor networks such as those found in smart-home, environmental monitoring, and industrial automation applications.

The SNAP/LE was also tunable, with the input voltage directly impacting the number of MIPS (here meaning "million instructions per second") attainable by the core. This range was significant, moving from 23MIPS at 0.6V to 200MIPS at 1.8V. While an interesting and commendable feat, this was not the most applicable trait to inject into the design, and the authors later released a similar platform with a more fine-tuned goal. The BitSNAP system focused entirely on the low-power angle of this technology. It was able to run at between 6-54MIPS and used only 24pJ per instruction, making it ideal for systems that are both self-powered and small-scale or intermittent in operation. The authors point out that while the MIPS range is far below a conventional processor, the application puts far higher weight on efficiency, and the ability to perform a huge number of different tasks is not always critical.

An interesting twist for BitSNAP was that it used a bit-serial datapath, which allows for flexible data input sizes. This is a difficult problem to solve in a synchronous environment, but is well-suited to asynchronous systems, since the datapath sizing component simple continues until it reaches the end of the description header and has no timing concerns with the larger system. This reuse of the same components is the motif that contributes to the smaller chip size of asynchronous designs overall.

B. GPS Processing

Like distributed sensors, the growing technology of imbedded GPS chips in many devices was a trend that drove more investigation into asynchronous applications. GPS requires a near-constant level of communication between an RF component and the processor which interpolates those signals. A solution was proposed which further subdivided the processing system in half. This split the processing tasks into a signal and sensor input processing which interfaced directly with the RF component and was able to take in real-time sensor inputs (handled optimally by an asynchronous core); and the triangulation and logical processing tasks, which are better suited to periodic calculation by a synchronous core.

The improvements achieved by this technique made continuous GPS signal reading a viable technology that has since become ubiquitous in hand-held devices.

IV. Neural Applications

After the success of asynchronous systems in the realm of conservation of power and size in the early 2000s, there has been a more recent technology that makes the paradigm influential once again. The concept of neural networks in the field of artificial intelligence refers to the software implementation of nodes that transform inputs to outputs. These nodes are often combined in large-scale webbings that produce sophisticated responses from simple building blocks given training data. This approach led researchers to consider mimicking the software implementation in hardware to optimize the performance of such AI code. Asynchronous systems are perfectly positioned to fill this niche – they are able to operate independently and have well-defined processes for inter-component communication that is fully scalable. There have been many approaches to implementing an asynchronous processor focused on neural network designs, and some that move past the software network and move to the inspiration: the human brain.

A. *DYNAPS*

Dynamic neuromorphic asynchronous processors (DYNAPS) are designed with the target to produce brain-inspired computers [5]. A key component of these neural networks is their event-based nature. While the inputs themselves contain information, the sequence and timings of those inputs are equally important and informational. This leads to an emphasis on the ability to always monitor all inputs and process any events exactly when they arrive. For high-speed interactions and inputs this requirement can mean that even very fast synchronous processors can miss information or make data wait for a component to be available.

To connect all the neurons in the network, conventional approaches tend to either use a tree-based technique with the goal of reducing used space and power consumption, or a full mesh that is flexible and connects all nodes. The former sacrifices speed and flexibility for efficiency, while the latter takes up a lot of resources to execute. Instead, the DYNAPS system uses a mixed mode hierarchical-mesh routing scheme that allows for the best of both worlds using custom asynchronous components that are quasi-delay insensitive.

All of these improvements were displayed by implementing a convolutional neural network (CNN) on the hardware that processed information directly from a visual sensor. By taking the input directly in and processing the events while factoring in their sub-clock timings, the network achieved substantial results with much lower power consumption than the traditional architecture.

B. *Digital Neurosynaptic Core*

Many approaches to modelling brain systems in hardware lean towards matching the incredible natural density of the mind. While this is great for the conceptual mimicking of neuron networks, these designs do not exist in a vacuum, and do not train and program themselves. For the human scientists that are using such dense systems, it is difficult or impossible to switch between conceptual models while using the same hardware. The development of a digital neurosynaptic core was targeted at providing a reconfigurable neural system that was both able to run in real-time while also being power-efficient [7]. The approach was to have a one-to-one correspondence between the software and hardware elements – neurons, axons, and synapses. This allows developers to iterate on algorithms using the software model independent of the hardware development. This can permit faster iteration of the overall product as an established interface will be maintained as in a traditional architecture

Unlike a traditional architecture, however, this hardware can process events in real-time without a gigahertz range clock that would consume an unscalable amount of power. This is accomplished by using a core that uses an asynchronous event-driven design similar to DYNAPS whose power consumption grows linearly with activity rather than a constant consumption driven by the clock rate. The benefit of the design of this neurosynaptic core is that it is extremely scalable and modular. Spikes from the neurons are processed in-place and do not need nearly as much communication between processor and memory as the same software implemented on conventional hardware such as a GPU.

C. *Brain-Computer Interfaces*

While the other examples discussed so far have attempted to mimic brain behavior and structure, brain-computer interfaces (BCIs) instead seek to effectively monitor, measure, and respond to real brain activity [8]. These interfaces are sometimes implemented by gathering signals from outside the skull of patients. These are the headsets of electrodes that are used to deduce brain activity using the signals emitted and measurable above the skin. This approach is not precise enough for proposed, cutting-edge brain therapies that require fine-tuned monitoring of actual tissue firings. Some BCIs, therefore, have been implanted within the skull and can both measure activity more accurately while also having the opportunity to directly influence the tissue through stimulation.

Current industry applications have been ASICs customized for illnesses, patient types, or a combination of both. With the understandable variety of neurological conditions, each personal to the patient, there is a demand for a programmable set of hardware backing a BCI. Reference [8] describes their hardware architecture for low-power (HALO) BCI system which utilizes asynchronous design to meet the stringent requirements for this type of processor.

While low power consumption is an appreciated benefit for much of computing, nowhere is it so critical as in BCIs. When power is consumed in hardware, it is dissipated as heat and electromagnetic radiation. There are understandably strict FDA regulations on the maximum heat dissipation that an implant can produce within the brain to avoid damage to the mounting tissue. For this reason, existing BCIs are often lower functionality, both in scope of tissue monitored and range of functions, to comply with the low power available. As been discussed, asynchronous design drastically increases how much is possible with a lower power processor. This allows a HALO BCI to achieve the same level of processing and algorithmic complexity as devices which are too intensive to currently implant in the brain itself

Further gains using HALO BCIs are seen in that the systems can contain enough functionality to be self-contained and closed loop. This is helpful in the numerous brain applications that involve both monitoring and intervening in neurological conditions. The paper gives a striking example of predicting seizure activity and mitigating the impact of that activity by stimulating the appropriate brain regions.

V. A Novel VR Application

To conclude, we will describe a potential additional niche use case for asynchronous processors: fully self-contained virtual reality (VR) wearables.

The conventional VR system consists of four main elements: the headset, controllers, spatial sensors, and the central processor. The headset provides the immersive viewing environment for the user, as well as a target for the sensors to detect and provides additional orientation information to the processor using accelerometers and other integrated sensors. The controllers serve a similar function with their onboard sensors and ability to be tracked by the external spatial sensors. The spatial sensors are typically mounted in the user's environment and take in visual input of the headset, controllers, and user's body which is also sent to the central processor. That processor is generally a high-powered PC which combines all the data sources to triangulate the exact position and orientation of the user and their controllers within the physical space. That spatial information is used to render the appropriate visuals which are then sent to the headset. Common implementations of this system are the Valve Index or the HTC Vive.

We propose a version of this system that combines all the components of a standard VR system into a single device that serves all the functions. The existing device which is closest to the proposed system is the Oculus Quest, a stand-alone VR system. The Quest is a VR headset which establishes its location through a combination of onboard accelerometers and a live feed of visual input from several cameras in the headset but does not need any external sensors or central processor. Instead, the Quest is able to process its own measured inputs, determine its location, render the visuals, and display them within the form factor of the headset.

This approach is very successful in lowering the barrier for experiencing VR – there is no complicated setup of sensors, or the purchase of an expensive powerful PC required to use it. That being said, there are several improvements that would increase the usability and global appeal of the device, and they are all improvements that asynchronous design can facilitate.

One such improvement would be the complete removal of controllers, relying instead on high fidelity hand-tracking. This is a feature that depends heavily on the real-time processing of high-quality visual inputs that must be mapped to lower-dimension predictions. These tasks are perfectly suited to the asynchronous solutions that have been described in this paper. We propose a split core processor, with a designated asynchronous processor that handles the constant input of video feeds and spatial sensor readings using a hierarchical-mesh design similar to the DYNAPS design to allocate a neuron to each pixel or similar unit of data coming in. A custom trained and architected neural net could then handle much of the heavy lifting for interpretation of the inputs. Dynamic movement requires a strong understanding of events' timing relative to each other, which is a strong point for the non-clocked implementation of sensor measurement.

A synchronous processor would then take in the filtered and mapped input data to run against the game engine. This would allow for development of third-party applications more easily since the underlying architecture for this subset of the device would be closer to industry standard. With the space saved by the smaller chip size of the asynchronous core, this GPU could be larger and present more vivid graphics to the user in a gaming or personal setting like the market that the Quest is currently targeting.

Alternatively, such a device could be used in a more utilitarian way. By instead keeping the graphics and application processor more reserved and minimalistic, the power consumption could be drastically reduced from current hardware. This would combine with the great benefit of an all-in-one VR wearable: spatial flexibility. While conventional systems restrain the users to the location in which they have their sensors arranged, a camera-based standalone system can be used anywhere. With the increased efficiency of an imbedded neural net in hardware, it may even be able to map spaces in near real-time and allow for a mixed-reality experience that could be used in nearly unlimited applications. For example, an interior designer can simply walk around a physical space and build a full CAD model of it, editing their design for the room as they move. Surgical training programs could be adapted dynamically to a patient as they arrive, indicating recommended procedure routes and locations.

The system could also take advantage of the ability of asynchronous hardware to manage large sensor networks with large amounts of inter-communication to create massively-multiplayer experiences between many users of the wearables at once. VR laser tag in fantastical locations could be possible, with live mapping of both the static environment and other users preventing collisions between players. Museum exhibits could gain an extra level of interactivity with crowds of onlookers simultaneously watching skeletons springing to life or following virtual tour guides through the physical halls.

To achieve all these features in a single device - power efficient, real-time neural network processing of event-based data, and large-scale sensor networking - is enabled by using asynchronous hardware designs to their potential. While their uses are still more niche compared to the synchronous processor paradigm, asynchronous designs have strengths that are extremely valuable in growing parts of technological advancement, and VR is a perfect example of the future impact that such hardware can have.

Acknowledgment

Special thanks to Professor Rajt Manohar of Yale University for his engaging courses on asynchronous hardware design and silicon compilation that sparked my interest in this topic, and for his contribution to many of the publications cited in this paper.

REFERENCES

[1] M. Tranchero and L. M. Reyneri, "Exploiting synchronous placement for asynchronous circuits onto commercial FPGAs," 2009 International Conference on Field Programmable Logic and Applications, 2009, pp. 622-625, doi: 10.1109/FPL.2009.5272378.

[2] Virantha Ekanayake, Clinton Kelly, and Rajit Manohar. 2004. An ultra low-power processor for sensor networks. In Proceedings of the 11th international conference on Architectural support for programming languages and operating systems (ASPLOS XI). Association for Computing Machinery, New York, NY, USA, 27–36. DOI:https://doi.org/10.1145/1024393.1024397 Q. Zhang and G. Theodoropoulos, "Modelling SAMIPS: a synthesisable asynchronous MIPS processor," 37th Annual Simulation Symposium, 2004. Proceedings., 2004, pp. 205-212, doi: 10.1109/SIMSYM.2004.1299484.

[3] V. N. Ekanayake, C. Kelly and R. Manohar, "BitSNAP: dynamic significance compression for a low-energy sensor network asynchronous processor," *11th IEEE International Symposium on Asynchronous Circuits and Systems*, 2005, pp. 144-154, doi: 10.1109/ASYNC.2005.14.

[4] S. B. Furber, D. A. Edwards and J. D. Garside, "AMULET3: a 100 MIPS asynchronous embedded processor," Proceedings 2000 International Conference on Computer Design, 2000, pp. 329-334, doi: 10.1109/ICCD.2000.878304.

[5] S. Moradi, N. Qiao, F. Stefanini and G. Indiveri, "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," in IEEE Transactions on Biomedical Circuits and Systems, vol. 12, no. 1, pp. 106-122, Feb. 2018, doi: 10.1109/TBCAS.2017.2759700.

[6] B. Z. Tang, S. Longfield Jr., S. A. Bhave and R. Manohar, "A Low Power Asynchronous GPS Baseband Processor," 2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems, 2012, pp. 33-40, doi: 10.1109/ASYNC.2012.20.

[7] J. V. Arthur et al., "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," The 2012 International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1-8, doi: 10.1109/IJCNN.2012.6252637.

[8] I. Karageorgos et al., "Hardware-Software Co-Design for Brain-Computer Interfaces," 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), 2020, pp. 391-404, doi: 10.1109/ISCA45697.2020.00041.

[9] A. Skaf, J. Simatic and L. Fesquet, "Seeking low-power synchronous/asynchronous systems: A FIR implementation case study," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1-4, doi: 10.1109/ISCAS.2017.8050379.