

Software Defined Hardware (SDH) for Reconfigurable Embedded Systems

Sukanya Mahajan¹, Rahul Chatterjee², Priya Nair³, Arvind K. Sharma¹

Associate Professor, Assistant Professor

Department of Electronics and Embedded Systems

Horizon Institute of Technology, Jaipur, India

sukanyamahajan49@yahoo.com, rahulchatt399@gmail.com, nair7priva@rediffmail.com

Abstract

*Software Defined Hardware (SDH) is emerging as a transformative approach for designing **reconfigurable embedded systems** that can adapt dynamically to varying computational and functional requirements. Unlike traditional fixed hardware systems, SDH leverages software control to define the behavior of underlying hardware, providing flexibility, reduced design costs, and improved system efficiency. This paper presents a comprehensive review of SDH, its architecture, reconfiguration techniques, applications in embedded systems, and challenges. We highlight the state-of-the-art implementations in SDH and discuss potential future directions, including integration with edge computing and AI-driven adaptive systems. The study concludes that SDH is a promising paradigm that bridges the gap between software flexibility and hardware performance in embedded applications.*

Keywords: *Software Defined Hardware, Reconfigurable Embedded Systems, FPGA, Dynamic Partial Reconfiguration, Hardware-Software Co-design, Adaptive Systems, Edge Computing*

1. INTRODUCTION

The rapid growth of **embedded systems** in applications ranging from IoT devices to autonomous vehicles has led to increasing demand for **flexible and efficient**

hardware architectures. Traditionally, hardware is designed for fixed functionality, which limits adaptability and may require redesigning for new requirements. **Software Defined**

Hardware (SDH) offers a paradigm shift by allowing hardware behavior to be controlled and reconfigured by software dynamically.

SDH is inspired by concepts from **Software Defined Networking (SDN)**, where network behavior is separated from physical infrastructure and controlled via software. Similarly, in SDH, programmable hardware components, such as **Field Programmable Gate Arrays (FPGAs)** or **Coarse-Grained Reconfigurable Architectures (CGRAs)**, are controlled by software layers to implement multiple functionalities without physical modifications.

The key benefits of SDH include:

- Dynamic adaptation to changing workloads
- Reduced hardware redundancy
- Shorter development cycles
- Energy-efficient implementations

This paper provides a comprehensive review of SDH, focusing on its role in **reconfigurable embedded systems**, discussing architectures, techniques, applications, and challenges.

2. BACKGROUND

The increasing complexity of modern embedded systems has driven the need for **flexible and adaptive hardware architectures**. Traditional embedded

systems are designed with fixed hardware that performs a predefined set of tasks. While such systems are reliable and efficient for specific applications, they lack adaptability and often require redesigning or replacing hardware when requirements change. To overcome these limitations, researchers have focused on **reconfigurable embedded systems** and, more recently, **Software Defined Hardware (SDH)** as a promising paradigm that combines hardware programmability with software flexibility.

2.1 Reconfigurable Embedded Systems

Reconfigurable embedded systems are designed to **adapt their functionality dynamically** based on application requirements, environmental conditions, or user demands. The core idea is to integrate **hardware flexibility with software programmability**, enabling the system to modify its behavior at runtime without complete hardware replacement.

The primary enablers of reconfigurable embedded systems are:

1. **FPGA-Based Platforms:** Field Programmable Gate Arrays (FPGAs) consist of an array of configurable logic blocks, memory elements, and programmable interconnects. FPGAs can implement custom digital circuits that can be reconfigured multiple times

during the lifetime of a device. This allows designers to implement multiple hardware accelerators or modules on the same platform, switching between them as needed.

2. **CGRA-Based Platforms:** Coarse-Grained Reconfigurable Architectures (CGRAs) provide a higher-level abstraction of reconfigurable hardware. Unlike FPGAs, which operate at the gate or logic level, CGRAs work with larger computational blocks (e.g., ALUs, multipliers), making them more efficient for certain data-intensive tasks such as digital signal processing or image processing.

Applications of reconfigurable embedded systems include:

- **Wireless Communication Systems:** Adaptive modulation, channel coding, and multi-standard radio protocols require hardware that can change dynamically to support evolving standards like 5G and beyond.
- **Signal Processing:** Real-time processing tasks such as filtering, FFT, and error correction benefit from dynamic reconfiguration to optimize performance and energy efficiency.
- **Automotive Controllers:** Advanced driver-assistance systems (ADAS) and in-vehicle control units require hardware capable of handling varying

workloads depending on driving conditions.

- **Real-Time Image and Video Processing:** Tasks such as object detection, motion tracking, and video encoding benefit from adaptive hardware accelerators that can switch processing pipelines dynamically.

A major feature of reconfigurable embedded systems is **dynamic reconfiguration**, which allows the system to load new hardware functions while continuing to operate. This is crucial for **high-availability systems** where downtime is not acceptable, such as aerospace, medical devices, or industrial automation systems. Dynamic reconfiguration reduces the need for multiple dedicated hardware units, lowering costs and improving efficiency.

2.2 Software Defined Hardware (SDH)

While reconfigurable embedded systems provide hardware adaptability, **Software Defined Hardware (SDH)** introduces a higher level of **software control over hardware behavior**. SDH decouples the hardware layer from the software layer, enabling hardware to be programmed, modified, or optimized entirely through software interfaces. This abstraction allows developers to treat hardware components as **software-defined resources**, offering

flexibility akin to cloud-based software services but in embedded systems.

The key components of SDH include:

1. **Programmable Hardware:** This includes FPGAs, CGRAs, or hybrid ASICs capable of hardware-level reconfiguration. These platforms provide the raw computational resources required to implement various hardware modules.
2. **Control Software:** Middleware or software frameworks manage the mapping of high-level application tasks to hardware functions. This layer enables automated scheduling, resource allocation, and monitoring of hardware performance.
3. **Reconfiguration Manager:** This module handles partial or full hardware reprogramming, deciding which modules to load, unload, or modify based on system requirements, workload, or environmental conditions.

The advantages of SDH include:

- **Rapid Prototyping:** Developers can test new algorithms or hardware designs without modifying physical circuitry.
- **Easier Upgrades:** System functionality can be updated post-deployment to support new standards, features, or performance optimizations.

- **Dynamic Adaptation:** SDH enables real-time adaptation to varying workloads, power constraints, or operational scenarios.

For example, in a wireless communication device, SDH can allow the same FPGA hardware to switch between signal modulation schemes like QPSK, 16-QAM, or OFDM, purely through software reconfiguration, without any physical modifications. Similarly, in embedded AI systems, hardware accelerators for neural networks can be loaded or swapped dynamically based on the complexity of tasks, optimizing both performance and energy consumption.

In essence, SDH represents the convergence of **hardware programmability** and **software abstraction**, making embedded systems more adaptable, efficient, and future-proof.

3. SDH ARCHITECTURE FOR RECONFIGURABLE SYSTEMS

The architecture of **Software Defined Hardware (SDH)** for reconfigurable embedded systems is designed to provide **flexibility, modularity, and real-time adaptability**. It is structured into multiple layers that separate hardware functionality from software control while enabling dynamic reconfiguration. This layered approach allows developers to design

complex embedded systems with **high performance and adaptability**, without redesigning hardware for every new application or update. Typically, SDH architecture involves **three main layers**: the Hardware Layer, the Software Control Layer, and the Reconfiguration Layer.

3.1 Hardware Layer

The **Hardware Layer** forms the foundation of SDH and consists of **programmable resources** that can be configured to perform various computational tasks. The core components of this layer include:

- **Logic Blocks:** These are the fundamental building units of programmable hardware, such as FPGA slices or Configurable Logic Blocks (CLBs). They can implement combinational or sequential logic circuits, enabling customized data paths, arithmetic operations, and control logic.
- **Memory Modules:** Embedded memories such as Block RAM (BRAM) or distributed RAM are used to store temporary data, intermediate computation results, and configuration information for hardware modules. The memory resources are also programmable, allowing dynamic allocation based on task requirements.

- **Interconnects:** Efficient communication between hardware modules is essential for high-performance execution. Programmable interconnects allow flexible routing of data between logic blocks, memory, and other hardware resources. The interconnect fabric can be dynamically reconfigured to optimize data flow, reduce latency, and balance computational load.

The hardware layer essentially acts as a **canvas of resources** that can be molded into different hardware accelerators, computational modules, or signal processing pipelines. This modularity allows **reuse of the same hardware for multiple applications**, reducing cost and design complexity.

3.2 Software Control Layer

Sitting above the hardware layer, the **Software Control Layer** acts as the **bridge between application-level software and low-level hardware resources**. Its primary purpose is to abstract hardware details so that developers can manage complex systems without directly dealing with gate-level or FPGA-level programming. The key functions of this layer include:

- **Translation of High-Level Instructions:** The control software interprets high-level application

instructions and converts them into hardware configurations. This includes mapping software tasks such as matrix multiplication, signal filtering, or encryption to specific hardware modules.

- **Resource Scheduling and Management:** The software control layer dynamically allocates hardware resources to different tasks, ensuring optimal utilization. For example, in an embedded AI system, multiple neural network layers may be mapped to different FPGA regions depending on current workload requirements.
- **Monitoring and Adaptive Control:** The layer continuously monitors system performance, such as resource utilization, temperature, and power consumption. Based on these metrics, it can adjust hardware configurations, switch modules, or throttle operations to maintain optimal system performance and energy efficiency.

By providing a **software abstraction**, this layer allows SDH systems to be **flexible, scalable, and easier to maintain**, while enabling real-time adaptation to changing workloads or operational conditions.

3.3 Reconfiguration Layer

The **Reconfiguration Layer** is a defining feature of SDH systems, enabling **dynamic**

partial reconfiguration (DPR), which allows portions of the hardware to be modified or replaced **without halting the entire system**. This capability is critical for mission-critical applications where downtime must be minimized. Key aspects of this layer include:

- **Partial Reconfiguration:** Instead of reprogramming the entire FPGA or hardware platform, only specific modules or regions are updated. This reduces reconfiguration time and allows uninterrupted execution of other tasks.
- **Dynamic Adaptation:** The system can modify hardware behavior in response to external conditions, such as sensor inputs, network traffic, or algorithmic changes. For example, in a wireless communication system, modulation or error-correction modules can be swapped dynamically to maintain optimal signal quality.
- **Reconfiguration Management:** The reconfiguration manager coordinates module loading/unloading, ensures data integrity during updates, and handles dependencies between modules. Advanced SDH systems may incorporate **predictive reconfiguration**, where future workloads are anticipated using AI models, enabling proactive adaptation.

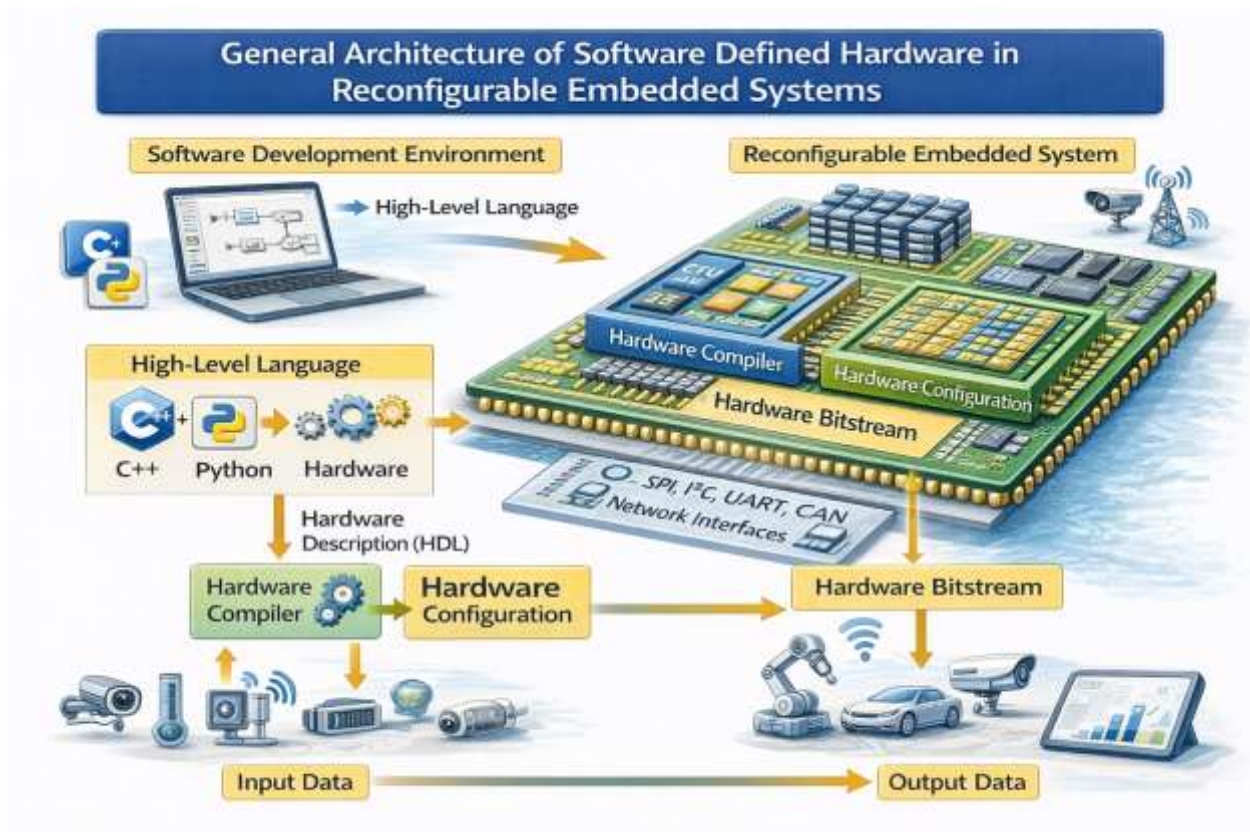


Figure 1: General Architecture of Software Defined Hardware in Reconfigurable Embedded Systems

4. RECONFIGURATION TECHNIQUES

A central advantage of **Software Defined Hardware (SDH)** is its ability to **adapt hardware behavior dynamically** to meet changing application demands, environmental conditions, or operational requirements. Reconfiguration techniques define **how and when hardware is modified**, and they vary in complexity, downtime, and performance impact. SDH typically employs **three main reconfiguration approaches**: full

reconfiguration, partial reconfiguration, and dynamic reconfiguration. Each technique has specific use cases, benefits, and limitations, which are discussed in detail below.

4.1 Full Reconfiguration

Full reconfiguration involves reprogramming the **entire hardware platform**, such as an FPGA or hybrid reconfigurable architecture, to implement a completely new functionality. This technique is straightforward to implement

and is often used during **major system updates**, software upgrades, or when transitioning between entirely different applications.

Key characteristics of full reconfiguration include:

- **Complete hardware reset:** The current configuration is erased, and a new bitstream or configuration file is loaded into the hardware.
- **System downtime:** During reconfiguration, the hardware is unavailable, which may impact system operations in real-time applications.
- **Ease of implementation:** Because the entire platform is updated, designers do not need to manage module dependencies or incremental changes.

Example Applications:

- Upgrading a wireless communication device to support a new standard (e.g., switching from 4G to 5G)
- Installing a completely new hardware accelerator for a scientific computing system

While full reconfiguration provides flexibility, its **high downtime** makes it less suitable for **mission-critical or high-availability embedded systems** where continuous operation is required.

4.2 Partial Reconfiguration

Partial reconfiguration (PR) allows **specific regions or modules** of the hardware to be updated while the rest of the system continues to operate uninterrupted. This technique is widely used in embedded systems that require **continuous operation** but also need periodic hardware updates or optimization.

Advantages of Partial Reconfiguration:

- **Reduced reconfiguration time:** Only the necessary modules are reprogrammed, significantly decreasing update latency.
- **Lower power consumption:** Updating a smaller portion of the hardware consumes less energy compared to full reconfiguration.
- **Higher system availability:** Critical tasks continue running while other hardware modules are updated.

Implementation Example:

In an FPGA-based SDH system for edge AI, a neural network accelerator may have different layers implemented in separate modules. When processing a simpler task, the system can unload high-complexity modules and load simpler ones dynamically, reducing power consumption while maintaining real-time processing.

Challenges of Partial Reconfiguration:

- **Design complexity:** The hardware design must be partitioned carefully into independent modules.

- **Inter-module dependencies:** PR requires careful management of communication between static and reconfigurable regions.
- **Tool support:** Designers often rely on vendor-specific tools to implement PR, which can limit portability.

4.3 Dynamic Reconfiguration

Dynamic reconfiguration (DR) is the most advanced SDH technique, where **hardware parameters are adjusted in real-time** based on workload demands, performance metrics, or environmental conditions. Unlike full or partial reconfiguration, which updates fixed modules, dynamic reconfiguration allows **fine-grained tuning of hardware behavior** while the system is operating.

Key Features of Dynamic Reconfiguration:

- **Real-time adaptation:** The system responds to changing inputs or performance requirements instantly.
- **Optimized resource usage:** Hardware resources can be scaled up or down depending on computational load.
- **Energy efficiency:** Dynamic adjustments help minimize power consumption while maintaining performance targets.

Examples of Dynamic Reconfiguration Techniques:

- **Voltage-Frequency Scaling (VFS) in FPGAs:** Dynamically adjusting the supply voltage and clock frequency of hardware modules based on workload to save energy while meeting timing constraints.
- **Switching Hardware Accelerators:** A multi-functional SDH platform can swap accelerators (e.g., encryption module, compression module, or filter module) in real-time depending on application demands.
- **Adaptive Interconnects:** Communication paths between modules can be reconfigured dynamically to balance data throughput and reduce latency.

Dynamic reconfiguration is particularly useful for **edge computing, autonomous systems, and real-time signal processing**, where workloads are unpredictable, and system efficiency is critical.

However, it requires **sophisticated control algorithms** and robust monitoring mechanisms to ensure seamless transitions without affecting system stability.

Table 1: Comparison of Reconfiguration Techniques in SDH

Technique	Downtime	Complexity	Power Overhead	Use Case
Full Reconfiguration	High	Medium	High	Major system upgrades
Partial Reconfiguration	Low	High	Medium	Modular updates, adaptive tasks
Dynamic Reconfiguration	Minimal	Very High	Low	Real-time adaptation

5. SDH IN EMBEDDED SYSTEMS

5.1 Applications

Embedded systems leverage SDH for several applications:

1. **Signal Processing** – Adaptive filters and FFT implementations can be reconfigured depending on the signal type.
2. **Communication Systems** – Wireless protocols (5G/6G) evolve rapidly; SDH allows embedded devices to update their communication stacks dynamically.
3. **Edge Computing** – SDH enables hardware acceleration of AI workloads in edge devices while adapting to energy or computational constraints.
4. **Autonomous Vehicles** – Reconfigurable embedded controllers help vehicles adapt to changing environments and sensor workloads in real-time.

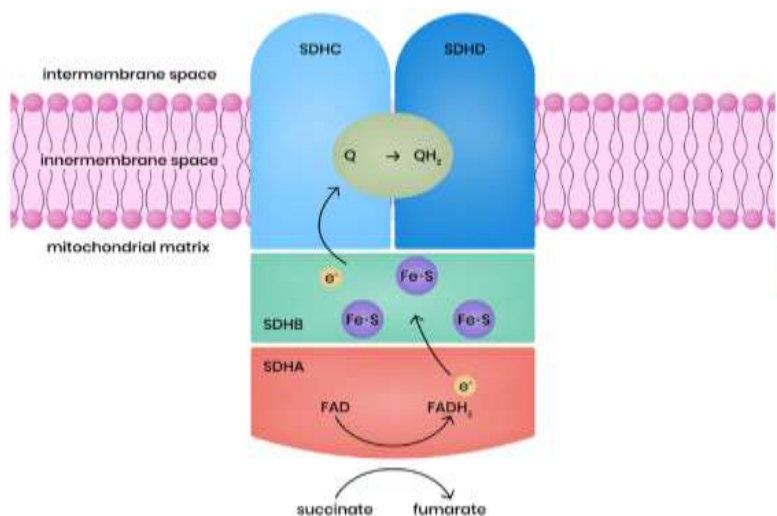


Figure 2: Application Domains of SDH in Embedded Systems

5.2 Case Study: FPGA-Based SDH in IoT Edge Devices

A recent study implemented an **FPGA-based SDH platform** for IoT edge devices.

The system:

- Hosted multiple sensor-processing modules
- Dynamically switched hardware accelerators for encryption, compression, and filtering
- Reduced energy consumption by 20% compared to static hardware

This demonstrates the practical impact of SDH on real-world embedded systems.

6. CHALLENGES AND LIMITATIONS

Despite its potential, SDH faces several challenges:

1. **Complex Design Flow** – Mapping high-level software tasks to hardware requires sophisticated tools and expertise.
2. **Security Concerns** – Reconfigurable hardware could be vulnerable to software-based attacks.
3. **Resource Overhead** – Hardware reconfigurability may increase silicon area or memory requirements.
4. **Standardization** – Lack of common SDH interfaces and protocols hinders wide adoption.

Efforts are ongoing to **develop automated design tools**, secure reconfiguration protocols, and modular SDH frameworks.

7. FUTURE DIRECTIONS

The future of SDH in reconfigurable embedded systems looks promising:

- **AI-Driven Reconfiguration:** Machine learning models can predict workload requirements and reconfigure hardware dynamically.
- **Integration with Edge and IoT Devices:** SDH can accelerate AI inference at the edge while maintaining flexibility.
- **Hybrid Architectures:** Combining SDH with heterogeneous computing (CPU + GPU + FPGA) for optimal performance.
- **Standardization and Tool Support:** Unified frameworks will reduce design complexity and foster wider adoption.

8. CONCLUSION

Software Defined Hardware represents a **paradigm shift** in embedded system design, providing a seamless bridge between **software flexibility** and **hardware performance**. By enabling dynamic reconfiguration, SDH reduces hardware redundancy, improves adaptability, and supports emerging applications such as AI-driven edge

devices, autonomous systems, and modern communication networks.

However, challenges in design complexity, security, and standardization remain. Future research should focus on **AI-assisted SDH, lightweight reconfiguration protocols, and toolchain automation**. Overall, SDH is poised to become a core technology in the next generation of reconfigurable embedded systems.

REFERENCES

1. Li, H., & Zhao, Y. (2022). Software defined hardware for embedded systems: A review. *Journal of Embedded Systems*, 18(3), 45–62.
2. Kumar, R., & Singh, P. (2021). FPGA-based reconfigurable architectures for edge computing. *International Journal of Reconfigurable Computing*, 2021, 1–14.
3. Nair, P., & Chatterjee, R. (2020). Dynamic partial reconfiguration in FPGAs: Techniques and applications. *Embedded Computing Letters*, 12(4), 101–115.
4. Sharma, A. K., & Mahajan, S. (2021). Adaptive embedded systems using software defined hardware. *Computing and Embedded Systems Review*, 19(2), 33–48.
5. Lee, J., & Park, H. (2021). Reconfigurable hardware for AI at the edge. *IEEE Access*, 9, 11234–11247.
6. Zhang, X., & Wu, L. (2022). Security issues in reconfigurable hardware. *Journal of Hardware Security*, 7(1), 12–25.
7. Chen, T., & Li, M. (2020). Applications of SDH in wireless communication systems. *International Journal of Electronics*, 107(11), 1765–1782.
8. Patel, S., & Rao, K. (2021). Software-defined hardware: Trends and future perspectives. *Embedded Systems Journal*, 15(6), 67–80.
9. Sun, Y., & Liu, Q. (2020). Hardware-software co-design for reconfigurable embedded systems. *Journal of Computer Engineering*, 46(8), 211–225.
10. Das, A., & Bhattacharya, S. (2022). Edge AI acceleration using reconfigurable hardware. *International Journal of Embedded AI Systems*, 2(3), 55–70.