# KM-DFX Reconfigurable Partition v3 Documentation

Vincent Vansant

December 21, 2023

# Contents

# 1 Introduction

Welcome to the documentation for the Reconfigurable Partition (RP) in the KM-DFX (Kernel-Managed Dynamic Function eXchange) system. This document serves as a guide for developers, system architects, and hardware enthusiasts who are venturing into the realm of adaptable hardware acceleration using Field-Programmable Gate Arrays (FPGAs).

## 1.1 The KM-DFX System: An Overview

The KM-DFX system stands at the forefront of hardware acceleration, merging the flexibility of software with the raw performance of hardware. It enables dynamic reconfiguration of FPGA partitions, allowing for on-demand hardware acceleration tailored to specific computational tasks. This system is particularly beneficial for applications requiring high-speed data processing, such as encryption, image processing, and complex algorithm computations.

## 1.2 The Role of Reconfigurable Partitions

At the heart of the KM-DFX system lies the RP – a versatile and dynamic segment of the FPGA. These partitions are designed to be adaptable, capable of loading various accelerators such as AES (Advanced Encryption Standard) engines, as per the software's requirements. The essence of these partitions is their ability to morph functionality seamlessly, offering unparalleled flexibility in hardware resource utilization.

## 1.3 Architectural Overview

Each RP in the KM-DFX system interfaces with the rest of the system through a well-defined set of interfaces:

- **Two AXI Stream Interfaces:** These interfaces cater to the input and output streams of an accelerator. They ensure efficient and high-speed data transfer, crucial for tasks requiring real-time processing.
- One **AXI Lite Interface:** This interface serves as the control channel for the accelerators. It provides a streamlined path for configuration and management, allowing precise control over the hardware acceleration processes.

## 1.4 IP for AXI Lite Interface

A key component of the KM-DFX system is the specially developed IP for the AXI Lite Interface. This IP is crafted to standardize the definition of control registers, providing a consistent interface for communication between the kernel and the hardware accelerators. Simplifying the integration and operation of various accelerators within the system.

## 1.5 Documentation Scope

This document provides a detailed look at the RPs in the KM-DFX system. It covers topics on how they connect to other parts of the system, how they're controlled through the AXI Lite Interface, and how to use them with different accelerators. The purpose is to help you understand what these parts can do so you can make the most of them for faster and more flexible hardware performance.

# 2 Interface Specifications

In Chapter 2, we focus on the critical interface specifications that define and facilitate communication between the static region of the KM-DFX system and each individual RP. Utilizing the AXI4 protocol, these interfaces — the AXI Stream and AXI Lite — ensure consistent and reliable interaction within the system. Additionally, we will touch upon the aspects of clock and reset, albeit briefly, as part of the overall interface discussion.

This chapter emphasizes the critical importance of matching the AXI4-based AXI Stream and AXI Lite interfaces between the static and RPs in the KM-DFX system. Ensuring compatibility is essential, as version mismatches could lead to errors. While the contents of each partition can evolve, these interfaces must remain consistent. We will also provide guidance on best practices for ensuring interface compatibility.

## 2.1 AXI Stream Interfaces

The RPs in the KM-DFX system are equipped with two critical AXI Stream interfaces, designed primarily to supply a continuous stream of data to the accelerators, emphasizing high throughput over latency minimization. One interface serves as the input (slave interface), and the other as the output (master interface). Both interfaces share identical AXI stream settings, including TREADY, TVALID, TLAST, TKEEP, and TDATA signals, which are integral for managing the data flow. A notable feature of these interfaces is their 64-bit data width, complemented by an 8-bit TKEEP, aligning with the system's focus on maximizing data throughput for enhanced performance in various applications.

### 2.1.1 Input (Slave) Interface

The input or slave interface is designed to receive data streams into the RP. Its configuration aligns with the standard AXI Stream protocol, ensuring smooth data intake from the processing system.

### 2.1.2 Output (Master) Interface

Conversely, the output or master interface handles the data streams exiting the RP. It follows the same protocol standards, providing a reliable pathway for data transmission back to the processing system.

### 2.1.3 Recommendations for Accelerator Integration

In our experience, integrating hardware accelerators with these interfaces often involves varying data width requirements. To accommodate this, we recommend using Xilinx's AXI Stream Data Width Converter IPs. These converters facilitate the adaptation of the standard 64-bit width to any specific width required by an accelerator.

Furthermore, for simplification, the use of AXI Stream Subset Converters can be beneficial to eliminate the TKEEP signal. This approach allows accelerators to process data without the need to handle TKEEP, streamlining the design and reducing complexity.

These recommendations aim to provide a flexible yet standardized approach to integrating various accelerators with the RPs.

## 2.2 AXI Lite Interface

The AXI Lite interface in the KM-DFX system is used for the configuration of hardware accelerators. This interface, with a data width of 32 bits and an address width of 16 bits, allows for a 64K address range, making it suitable for a variety of configuration tasks.

### 2.2.1 General Configuration

Primarily, the AXI Lite interface is utilized for standard configuration processes. For this purpose, we have developed a specialized AXI REGISTER IP, tailored to fit these needs. The detailed functionality and implementation of this IP are covered in a later chapter.

### 2.2.2 Advanced Applications

In scenarios requiring substantial memory, the AXI Lite interface's versatility becomes evident. For instance, when implementing a RISC-V Ibex core, this interface was effectively used to connect with the core's dual-port BRAM. Such specific applications demonstrate the interface's adaptability to different operational needs.

## 2.3 Clock and Reset

Each RP within the KM-DFX system is allocated a single clock and a single reset signal. These signals are fundamental to maintaining synchronous operation across the partition, as they are routed uniformly to every module within the RP. This design ensures that all components within a partition operate in sync, avoiding complexities associated with asynchronous clocking.

### 2.3.1 Clock Distribution

It is important to note that within the confines of an RP, converting this clock to different frequencies is generally not supported, as per Xilinx's current capabilities and guidelines. This limitation emphasizes the importance of careful clock management and planning during the design phase of each module within the RP.

### 2.3.2 Reset Mechanism

The reset signal is uniformly distributed to all modules, facilitating a coordinated reset process across the partition. In addition to this primary hardware reset, there is a secondary reset mechanism specifically for stream modules and accelerators (excluding the AXI Lite interface). This secondary reset, originating from the AXI REGISTERS IP, enables a software-triggered reset. This functionality is for scenarios where an error occurs during operation or when reprogramming the partitions, allowing for a reset to be initiated via software.

# 3 AXI REGISTER IP

In Chapter 3, we focus on the AXI REGISTER IP, an integral component in the KM-DFX system, primarily developed to standardize communication between the kernel and hardware accelerators.

The AXI REGISTER IP is not only equipped with a set of default configurations, which are highly adaptable to meet the extensive configuration needs of various accelerators, but it also allows for complete customization. This flexibility enables the IP to cater to a wide range of requirements - from basic to highly specialized. In the case a users is seeking to implement more complex solutions, such as integrating large memory blocks with BRAM, they have the option to develop entirely custom IPs. However, it is important to note that opting for a custom approach means engaging with the full complexity of the system, which demands a deeper understanding and careful handling and may not be supported by the kernel.

## 3.1 General Configurations of the AXI REGISTER IP

In this subsection, we explore the general configurations of the AXI REGISTER IP within the KM-DFX system. The system's versatility is embodied in its range of configurations, each tailored to different levels of functionality and complexity. The AXI REGISTER IP is available in three primary configurations with the next always being a super-set of the previous:

- **Lite Configuration** Targeted for basic operational needs, the Lite Configuration is streamlined for simplicity and cost-effectiveness. It encompasses essential registers necessary for standard operations, making it an ideal choice for accelerators that need minimal configuration.
- **Medium Configuration** Expanding on the Lite Configuration, the Medium Configuration introduces additional registers for enhanced performance monitoring and control. This setup is well-suited for applications demanding more intricate operational data and a higher level of control features, beyond the basics. It also allows for an expanded range of user-defined registers.
- **Extended Configuration** At the top end of the spectrum lies the Extended Configuration, offering the most comprehensive feature set. It includes advanced monitoring registers and an expanded range of user-defined registers.

The configurations within the AXI REGISTER IP are strategically designed to cater to varying degrees of future-proofing. The Extended Configuration, in particular, stands out for its extensive performance monitoring capabilities. It paves the way for potential enhancements in the KM-DFX system, extending beyond mere kernel control to potentially encompass full Operating System (OS) level management. This could parallel the control that an OS exerts over processes and threads, such as allocating them across different cores. Such an expansion would significantly elevate the system's versatility, allowing for more dynamic and efficient allocation of tasks and resources within the KM-DFX environment. This forward-thinking approach ensures that the system remains adaptable and relevant in evolving computational landscapes.

## 3.2 Read and Read/Write Register Distinction in AXI REGISTER IP

In this subsection, we focus on the differentiation between read and read/write registers in the AXI REGISTER IP, a distinction that significantly simplifies the hardware interface within the KM-DFX system.

### 3.2.1 Perspective and Naming Convention

The classification of registers is from the perspective of the Processing System (PS). All registers in the system are readable, but only the read/write registers are writable. In the IP, read ports are labeled as INPUT_REG, while read/write ports are denoted as OUTPUT_REG. If the configuration of the IP is set to medium or extended two more ports become available for accelerator specific needs named EXTRA_OUTPUT_REG and EXTRA_INPUT_REG.

### 3.2.2 Read Registers (INPUT_REG)

Read registers are designed to be written to by the hardware. To streamline their operation, we have eliminated the need for a read enable signal. As a result, these registers are updated every clock cycle with the data provided by the hardware at that moment.

#### 3.2.3 Read/Write Registers (OUTPUT_REG)

Conversely, read/write registers are accessible for reading by the hardware but can be written to only by the PS. Any updates made to these registers from the PS side are reflected in the hardware output lines in the subsequent clock cycle. This arrangement allows for controlled modifications to the system's behavior based on the PS inputs, without the need for intricate synchronization mechanisms.

#### 3.2.4 Simplified Communication and Race Condition Avoidance

This subdivision into read and read/write registers serves a crucial purpose: it streamlines the communication process between the partition and the kernel. By establishing a one-way communication path for each register type, the system effectively avoids potential race conditions. The hardware interface remains straightforward, focusing on either providing data to the PS or receiving instructions from it, without the complexity of handling bidirectional data flows in a single register.

### 3.3 Register Mapping and Definitions

In this subsection, we will outline the structure and mapping of registers within the AXI REGISTER IP, crucial for understanding how the KM-DFX system manages data flow and configurations.

#### 3.3.1 Register Mapping Explained

Each register in the AXI REGISTER IP is 32 bits in size. However, due to the presence of a strobe signal in the AXI Lite interface, there is the technical capability to handle data with a granularity of 8 bits. The system utilizes a 64K address space, with each register occupying four addresses.

#### 3.3.2 Order and Priority of Addresses

The mapping of addresses follows a specific order:

1. **Read Registers:** These are placed at the beginning of the address space. By prioritizing read registers, we ensure that critical registers like the ID register and the configuration register are always located at consistent and predictable addresses. This strategic placement is instrumental for the kernel to identify which partition is loaded and to understand its configuration, solely based on the hardware information.
2. **Read/Write Registers:** Following the read registers, the read/write registers are mapped. These are primarily used for sending data or instructions from the PS to the RP.
3. **Additional Registers for Medium/Extended Configurations:** In cases where the medium or extended configurations are used, additional read registers and read/write registers are included. These are mapped after the standard read and read/write registers and are designed to cater to more complex operational requirements per user definition.

#### 3.3.3 Importance of Mapping Order

The reason for this specific ordering is to facilitate ease of access and predictability. By always having the ID and configuration registers at a fixed location, the system ensures straightforward identification and configuration processes.

### 3.4 Identification and Versioning Registers

This section details the Identification Registers. These registers are fundamental for the proper identification and management of hardware accelerators. Consisting of the Device ID Register and the Version Register, they provide information necessary for system initialization, compatibility checks, and firmware management. Their design ensures that the software can accurately recognize the hardware and its capabilities, as well as track firmware versions for maintenance and updates.

### 3.4.1 Device ID Register (READ) (LITE)

The Device ID Register is designed to uniquely identify the accelerator hardware. This register is structured as:

```
 ------------------------------------------------------------------------------------------------
| Device ID (32 bits)                                                                            |
 ------------------------------------------------------------------------------------------------
| Bits 31.....................................................................................0 |
 ------------------------------------------------------------------------------------------------
```

- **Device ID (Bits 31-0):** This 32-bit identifier is uniquely assigned to the specific type or model of the accelerator, crucial for software recognition and compatibility assurance.

**Address Definition:**

- **Address LITE:** 0x00
- **Address MEDIUM:** 0x00
- **Address EXTENDED:** 0x00

### 3.4.2 Version Register (READ) (LITE)

The Version Register stores the firmware version of the accelerator hardware. This is structured as:

```
 ------------------------------------------------------------------------------------------------
| Version (32 bits)                                                                              |
 ------------------------------------------------------------------------------------------------
| Bits 31.....................................................................................0 |
 ------------------------------------------------------------------------------------------------
```

- **Version (Bits 31-0):** Occupying the entire 32 bits, this register represents the version number of the accelerator's firmware or hardware, facilitating version tracking for maintenance and updates.

**Address Definition:**

- **Address LITE:** 0x04
- **Address MEDIUM:** 0x04
- **Address EXTENDED:** 0x04

## 3.5 Control and Management Registers

### 3.5.1 Configuration Register (READ) (LITE)

The Configuration Register defines the operational parameters of the accelerator. This register is structured as:

```
 ------------------------------------------------------------------------------------------------
| Input Width (8 bits) | Output Width (8 bits) | Reg. Set Config (8 bits) | Reserved (8 bits) |
 ------------------------------------------------------------------------------------------------
| Bits 31...........24 | Bits 23............16 | Bits 15................8 | Bits 7.........0 |
 ------------------------------------------------------------------------------------------------
```

- **Input Width (Bits 31-24):** These 8 bits specify the input data width in bytes for the AXI Stream interface. This width determines the amount of data the accelerator can receive in a single transaction.
- **Output Width (Bits 23-16):** Similar to the input width, these 8 bits define the output data width in bytes for the AXI Stream interface.

- **Register Set Configuration (Bits 15-8):** This section is used to specify the configuration mode of the accelerator:
  - **0x00 - Lite:** Indicates the Lite configuration, providing basic functionality with a minimal set of registers.
  - **0x01 - Medium:** Selects the Medium configuration, offering additional registers for performance monitoring and control.
  - **0x02 - Extended:** Activates the Extended configuration, which includes a comprehensive set of registers for advanced monitoring and control.
- **Reserved (Bits 7-0):** These 8 bits are reserved for future use or additional configuration parameters.

**Address Definition:**

- **Address LITE:** 0x08
- **Address MEDIUM:** 0x08
- **Address EXTENDED:** 0x08

### 3.5.2 Control Register (READ/WRITE) (LITE)

The Control Register is integral to control primary functions of the accelerator. This register is structured as:

```
 ------------------------------------------------------------------------------------------------
| Reserved (Bits 31-6)                          | Predefined Control Registers (Bits 5-0)      |
 ------------------------------------------------------------------------------------------------
| Bits 31.....................................6 | Bits 5.....................................0 |
 ------------------------------------------------------------------------------------------------
```

- **Start/Stop (Bit 0):** Toggles the operation of the accelerator. Writing a '1' starts or resumes operation; writing a '0' stops it.
- **Reset (Bit 1):** Resets the accelerator to its initial state, useful for initialization or recovery.
- **Performance Mode (Bit 3):** Switches between different operational modes, such as energy-saving or high-performance modes.
- **Debug Mode (Bit 4):** Activates debug mode for diagnostics and troubleshooting.
- **Configuration Lock (Bit 5):** Locks the configuration settings, preventing unintended changes during operation.
- **Reserved (Bits 6-31):** Reserved for future use or additional control features.

**Address Definition:**

- **Address LITE:** 0x18
- **Address MEDIUM:** 0x30
- **Address EXTENDED:** 0x70

### 3.5.3 Additional Read/Write Registers Configuration Register (READ) (MEDIUM)

This register indicates the number of additional read and write registers available. This register is structured as:

```
 ---------------------------------------------------------------------------------------------
| Additional Read Registers (Bits 31-16)       | Additional Write Registers (Bits 15-0)       |
 ---------------------------------------------------------------------------------------------
| Bits 31.................................16 | Bits 15.................................0 |
 ---------------------------------------------------------------------------------------------
```

- **Upper 16 Bits (Bits 31-16):** Define the number of additional read registers available. This allows for flexibility in extending the accelerator's functionalities.
- **Lower 16 Bits (Bits 15-0):** Specify the number of additional read/write registers available. This allows for flexibility in extending the accelerator's functionalities.

**Address Definition:**

- **Address MEDIUM:** 0x14
- **Address EXTENDED:** 0x14

## 3.6 Status and Monitoring Registers

### 3.6.1 Status Register (READ) (LITE)

The Status Register provides information about the current state of the accelerator. This register is structured as:

```
 ---------------------------------------------------------------------------------------------
| User-Defined Status (Bits 31-16)             | Predefined Status (Bits 15-0)                |
 ---------------------------------------------------------------------------------------------
| Bits 31.................................16 | Bits 15.................................0 |
 ---------------------------------------------------------------------------------------------
```

- **User-Defined Status (Bits 31-16):** The upper 16 bits are reserved for user-defined status indicators. This allows for custom status monitoring tailored to specific applications or operational requirements.
- **Predefined Status (Bits 15-0):** The lower 16 bits are allocated for predefined status indicators, which include:
  - **Bit 0 - Ready:** Indicates whether the accelerator is ready to receive new tasks or data.
  - **Bit 1 - Busy:** Shows if the accelerator is currently processing tasks.
  - **Bit 2 - Idle:** Indicates that the accelerator is idle and not processing any tasks.
  - **Bit 3 - Error:** Flags if there is an error; details can be found in the error code register.
  - **Bit 4 - Data Available:** Signifies that data is available for reading from the accelerator.
  - **Bit 5 - Task Complete:** Indicates the completion of a task.
  - **Bit 6 - Power Mode:** Reflects the current power mode or state of the accelerator.
  - **Bit 7 - Maintenance Mode:** Shows if the accelerator is in maintenance or diagnostic mode.
  - **Bits 8-15:** Reserved for future specific status types or additional functionality.

**Address Definition:**

- **Address LITE:** 0x0C
- **Address MEDIUM:** 0x0C
- **Address EXTENDED:** 0x0C

### 3.6.2 Status Clear Register (READ/WRITE) (LITE)

The Status Clear Register is used to acknowledge and reset specific status flags in the Status Register. This allows for precise control over the status indicators and helps maintain an accurate representation of the accelerator's current state. This register is structured as:

```
 ---------------------------------------------------------------------------------------------
| User-Defined Status Clear (Bits 31-16)       | Predefined Status Clear (Bits 15-0)          |
 ---------------------------------------------------------------------------------------------
| Bits 31.................................16 | Bits 15.................................0 |
 ---------------------------------------------------------------------------------------------
```

- **User-Defined Status Clear (Bits 31-16):** These bits allow the software to reset user-defined status indicators. Writing a '1' to a specific bit acknowledges and clears the corresponding user-defined status flag.
- **Predefined Status Clear (Bits 15-0):** These bits correspond to the predefined status indicators in the Status Register:
  - **Bit 0 - Ready:** Clear the 'Ready' status indicator.
  - **Bit 1 - Busy:** Reset the 'Busy' status flag.
  - **Bit 2 - Idle:** Acknowledge and clear the 'Idle' status.
  - **Bit 3 - Error:** Clear the 'Error' status after handling the error condition.
  - **Bit 4 - Data Available:** Reset the 'Data Available' indicator once data is read.
  - **Bit 5 - Task Complete:** Clear the 'Task Complete' status after acknowledging task completion.
  - **Bit 6 - Power Mode:** Use to reset or change the 'Power Mode' status.
  - **Bit 7 - Maintenance Mode:** Clear the 'Maintenance Mode' indicator once maintenance or diagnostics are completed.
  - **Bits 8-15:** Reserved for future specific status types or additional functionalities.

**Address Definition:**

- **Address LITE:** 0x1C
- **Address MEDIUM:** 0x34
- **Address EXTENDED:** 0x74

### 3.6.3 Error Code Register (READ) (LITE)

The Error Code Register is designed to indicate the types of errors encountered by the accelerator. This register is structured as:

```
 ---------------------------------------------------------------------------------------------
| User-Defined Error Codes (Bits 31-16)        | Predefined Error Codes (Bits 15-0)           |
 ---------------------------------------------------------------------------------------------
| Bits 31.................................16 | Bits 15.................................0 |
 ---------------------------------------------------------------------------------------------
```

- **User-Defined Error Codes (Bits 31-16):** The upper 16 bits are allocated for user-defined error codes, allowing for customized error tracking specific to different applications or use-cases.
- **Predefined Error Codes (Bits 15-0):** The lower 16 bits contain predefined error codes that cover common error scenarios in the accelerator's operation:
  - **Bit 0 - General Failure:** Indicates a generic, unspecified error.
  - **Bit 1 - Communication Error:** Signals issues related to communication interfaces.
  - **Bit 2 - Data Integrity Error:** Points to errors in data corruption or invalidity.

- **Bit 3 - Resource Overrun:** Indicates an overrun of the accelerator's resources.
- **Bit 4 - Timeout Error:** Triggered when a process or response exceeds the expected time limit.
- **Bit 5 - Configuration Error:** Reflects errors in configuration settings.
- **Bit 6 - Access Violation:** Occurs with unauthorized or inappropriate access attempts.
- **Bit 7 - Unsupported Operation:** Activated for invalid or unsupported functions.
- **Bits 8-15:** Reserved for future specific error types or additional functionality.

**Address Definition:**

- **Address LITE:** 0x10
- **Address MEDIUM:** 0x10
- **Address EXTENDED:** 0x10

### 3.6.4 Performance Counter Register (READ) (MEDIUM)

The Performance Counter Register is dedicated to tracking the number of tasks or operations completed by the accelerator. This register is structured as:

```
 ------------------------------------------------------------------------------------------------
| Performance Counter (32 bits)                                                                  |
 ------------------------------------------------------------------------------------------------
| Bits 31.....................................................................................0 |
 ------------------------------------------------------------------------------------------------
```

- **Performance Counter (Bits 31-0):** Occupying the entire 32 bits, this register increments with each task completed, offering a real-time metric of the accelerator's productivity. A task is yet to be defined.

**Address Definition:**

- **Address MEDIUM:** 0x18
- **Address EXTENDED:** 0x18

### 3.6.5 Data Rate Monitor Register (READ) (MEDIUM)

The Data Rate Monitor Register is used to monitor the data processing rate of the accelerator. This register is structured as:

```
 ------------------------------------------------------------------------------------------------
| Data Rate Monitor (32 bits)                                                                    |
 ------------------------------------------------------------------------------------------------
| Bits 31.....................................................................................0 |
 ------------------------------------------------------------------------------------------------
```

- **Data Rate Monitor (Bits 31-0):** Occupying the entire 32 bits, this register stores values indicating the current data rate, such as bytes per second, enabling real-time monitoring of data throughput. Data Rate is yet to be defined.

**Address Definition:**

- **Address MEDIUM:** 0x1C
- **Address EXTENDED:** 0x1C

### 3.6.6 Average Throughput Register (READ) (EXTENDED)

The Average Throughput Register is used to monitor the average data throughput of the accelerator over a defined period. This register is structured as:

```
 ------------------------------------------------------------------------------------------------
| Avg Throughput (32 bits)                                                                       |
 ------------------------------------------------------------------------------------------------
| Bits 31.....................................................................................0 |
 ------------------------------------------------------------------------------------------------
```

- **Avg Throughput (Bits 31-0):** Occupying the entire 32 bits, this register calculates the average throughput, aiding in performance analysis and optimization. Throughput and time period yet to be defined.

**Address Definition:**

- **Address EXTENDED:** 0x20

### 3.6.7 Peak Throughput Register (READ) (EXTENDED)

The Peak Throughput Register records the highest data throughput rate observed by the accelerator. This register is structured as:

```
 ------------------------------------------------------------------------------------------------
| Peak Throughput (32 bits)                                                                      |
 ------------------------------------------------------------------------------------------------
| Bits 31.....................................................................................0 |
 ------------------------------------------------------------------------------------------------
```

- **Peak Throughput (Bits 31-0):** Occupying the entire 32 bits, this register calculates the Peak throughput, aiding in performance analysis and optimization. Throughput and time period yet to be defined.

**Address Definition:**

- **Address EXTENDED:** 0x24

### 3.6.8 Resource Utilization Register (READ) (EXTENDED)

The Resource Utilization Register gives insights into the usage of critical resources within the accelerator, such as memory. This register is structured as:

```
 ------------------------------------------------------------------------------------------------
| Resource Utilization (32 bits)                                                                 |
 ------------------------------------------------------------------------------------------------
| Bits 31.....................................................................................0 |
 ------------------------------------------------------------------------------------------------
```

- **Resource Utilization (Bits 31-0):** Occupying the entire 32 bits, this register provides a snapshot of current resource usage, for optimizing performance and avoiding bottlenecks. Exact usage is to be defined.

**Address Definition:**

- **Address EXTENDED:** 0x28

## 3.7 Customization and Extension Registers

### 3.7.1 User-Defined Data Register (READ/WRITE) (LITE)

The User-Defined Data Register provides a straightforward way for users to send and store custom data. This register is particularly useful for applications that require only minimal data transfer or storage capabilities. This register is structured as:

```
 ------------------------------------------------------------------------------------------------
| User Data (32 bits)                                                                            |
 ------------------------------------------------------------------------------------------------
| Bits 31.....................................................................................0 |
 ------------------------------------------------------------------------------------------------
```

- **User Data (Bits 31-0):** Allows users to send or store 32 bits of custom data, suitable for simple control signals, status information, or configuration data.

**Address Definition:**

- **Address LITE:** 0x20
- **Address MEDIUM:** 0x38
- **Address EXTENDED:** 0x78

### 3.7.2 Additional User-Defined Data Registers (READ AND READ/WRITE) (MEDIUM)

The Additional User-Defined Data Registers allow for a user specified amount to store read or write data. The user can select however many extra read and read/write registers they need and these will be appended to all other registers (first read then write). The size of these should also be passed to the "Additional Read/Write Registers Configuration Register". These registers are structured as:

```
 ------------------------------------------------------------------------------------------------
| User READ Data 1 (32 bits)                                                                     |
 ------------------------------------------------------------------------------------------------
| ....                                                                                           |
 ------------------------------------------------------------------------------------------------
| User READ Data N (32 bits)                                                                     |
 ------------------------------------------------------------------------------------------------

 ------------------------------------------------------------------------------------------------
| User READ/WRITE Data 1 (32 bits)                                                               |
 ------------------------------------------------------------------------------------------------
| ....                                                                                           |
 ------------------------------------------------------------------------------------------------
| User READ/WRITE Data N (32 bits)                                                               |
 ------------------------------------------------------------------------------------------------
```

- **User READ Data (Bits 31-0):** Can be any data this specific accelerator needs.
- **User READ/WRITE Data (Bits 31-0):** Can be any data this specific accelerator needs.

**Address Definition READ:**

- **Address MEDIUM:** 0x4C - 0x4C+(N-1)*4
- **Address EXTENDED:** 0xAC - 0xAC+(N-1)*4

**Address Definition READ/WRITE:**

- **Address MEDIUM:** 0x4C+(N-1)*4 - 0x4C+(N-1)*4+(M-1)*4
- **Address EXTENDED:** 0xAC+(N-1)*4 - 0xAC+(N-1)*4+(M-1)*4

### 3.7.3 Empty Registers (READ AND READ/WRITE) (LITE)

The Empty Registers in the KM-DFX system are reserved 32-bit spaces intended for future use or potential expansions of the accelerator's functionalities. These registers are currently unassigned but provide a buffer for future enhancements or additional features that may be required as the system evolves. The number of these registers varies depending on the configuration of the accelerator:

```
 ---------------------------------------------------------------------------------------------------
|  Empty Reserved READ 1 (32 bits)                                                                  |
 ---------------------------------------------------------------------------------------------------
| ....                                                                                              |
 ---------------------------------------------------------------------------------------------------
| Empty Reserver READ 1 (LITE) or 4 (MEDIUM) or 16 (EXTENDED) (32 bits)                             |
 ---------------------------------------------------------------------------------------------------

 ---------------------------------------------------------------------------------------------------
|  Empty Reserved READ/WRITE 1 (32 bits)                                                            |
 ---------------------------------------------------------------------------------------------------
| ....                                                                                              |
 ---------------------------------------------------------------------------------------------------
| Empty Reserver READ/WRITE 1 (LITE) or 4 (MEDIUM) or 16 (EXTENDED) (32 bits)                       |
 ---------------------------------------------------------------------------------------------------
```

- **Empty Reserved READ (Bits 31-0):** Placeholders for future functionalities that are not currently defined.
- **Empty Reserved READ/WRITE (Bits 31-0):** Placeholders for future functionalities that are not currently defined.

**Address Definition READ:**

- **Address LITE:** 0x14
- **Address MEDIUM:** 0x20 - 0x2C
- **Address EXTENDED:** 0x30 - 0x6C

**Address Definition READ/WRITE:**

- **Address LITE:** 0x24
- **Address MEDIUM:** 0x3C - 0x48
- **Address EXTENDED:** 0x7C - 0xA8

## 3.8 Power Management Registers

### 3.8.1 Power Mode Register (READ) (EXTENDED)

The Power Mode Register that monitors the power mode/state of the accelerator. This register is structured as:

```
 ---------------------------------------------------------------------------------------------------
| Power Mode (32 bits)                                                                              |
 ---------------------------------------------------------------------------------------------------
| Bits 31.....................................................................................0 |
 ---------------------------------------------------------------------------------------------------
```

- **Power Mode (Bits 31-0):** Occupying the entire 32 bits, this register represents the current power management of the system. Exact definition yet to be defined.

**Address Definition:**

- **Address EXTENDED:** 0x2C

# 4 Future Developments

## 4.1 Expanding and Refining Performance Metrics

In the current state of the KM-DFX system, some performance metrics registers have yet to be clearly defined in terms of their specific measurement criteria. As the system evolves and its applications become more varied and complex, the need for additional registers with well-defined performance metrics will likely emerge. Future development efforts could focus on:

- **Detailed Performance Metrics:** Establishing clearer definitions for existing performance metrics, ensuring that each register provides valuable and precise data relevant to the accelerator's operation.
- **Additional Metrics Registers:** Introducing new registers to capture a broader range of performance data, such as real-time efficiency, latency measurements, and detailed resource utilization metrics.

## 4.2 Interrupt Capabilities

A significant area for future improvement is the implementation of interrupt capabilities originating from the partitions. This development would involve:

- **Hardware Integration:** Incorporating additional hardware within each partition to support interrupt generation and handling.
- **New Interface Development:** Establishing a new interface between the RPs and the static region to manage these interrupts effectively. Hence implementing this not only requires a new RP version but also a corresponding KM-DX upgrade.

## 4.3 Improved Support for BRAM Utilization

One of the key areas for future development in the KM-DFX system is improving support for the use of Block RAM (BRAM) in situations where larger memory is required, beyond what is practical with standard registers. This improvement would involve:

- **Documented Guidelines for BRAM Integration:** Developing comprehensive documentation on how to effectively integrate BRAM into the system. This would include best practices, design patterns, and examples to guide users in leveraging BRAM for memory-intensive applications.
- **Tools and Interfaces for BRAM Management:** Creating specialized tools or interfaces that facilitate the seamless integration and management of BRAM within the reconfigurable partitions. This would make it easier for users to implement memory solutions that are more extensive than what the current register-based approach allows.
- **Optimizing BRAM Accessibility:** Ensuring that BRAM can be accessed and managed efficiently, with minimal overhead.

## 4.4 Openness to Community Contributions

The KM-DFX system is a living project, and we encourage community involvement. Feedback and collaborative efforts from the user community are invaluable for refining existing features and introducing new functionalities.

# 5 Acknowledgements

As we bring this documentation to a close, I would like to show my appreciation to those who have helped in the development of this documentation.

Firstly, I'd like to acknowledge Eshra Ahmed, Jalap Hassin, Luo Geng, and Ma Yuhao for their dedication and tireless efforts. Their contributions have been crucial in creating and testing the KM-DFX system's Reconfigurable Partition.

I also extend my thanks to the teaching team – Josep Balasch, Yuri Cauwerts, and Songqiao Cui. Their support and wisdom, particularly Josep Balasch's role in shaping the AXI REGISTER IP, have been essential.

A special thanks to Ronny Webers, whose guidance steered me in the right direction for creating the AXI REGISTER IP.

Last but certainly not least, I extend my gratitude to Johan Vansant for reading the first draft of this documentation. His feedback was helpful in refining not only the structure of the text but also in providing notes on potential future enhancements, such as the interrupt capabilities.

Thank you all for your contributions.