AN13: Handling Time-Multiplexed (TM) Signals in SciCompiler

Abstract

In high-performance data acquisition systems such as the V2730 board, signals are provided in a Time-Multiplexed (TM) format to handle sampling rates exceeding 200 MHz. This application note demonstrates how to use SciCompiler IP blocks—specifically **TM SPLITTER** and **TM PACKER**—to split the TM data into parallel bit-vector signals, apply a scaling factor (less than 1) to each slice, and then reassemble the TM signal for further DSP operations.

https://github.com/NuclearInstruments/sci-compiler-time-mux

1 Introduction

In high-performance data acquisition systems, such as those based on the **V2730 board**, sampling rates can exceed 200 MHz. In these scenarios, instead of providing one sample per clock cycle, the hardware delivers multiple samples in parallel at a lower clock frequency. For example, the V2730 provides four **16-bit** samples simultaneously at **125 MHz**, achieving an effective sampling rate of **500 MSPS** (Mega Samples Per Second). This parallel data structure is referred to as **Time-Multiplexed (TM) data** in SciCompiler.

SciCompiler introduces **TM signals** to handle these parallel time slices. Each TM signal is essentially an **array of bit vectors**, with one element per time slice. For the V2730, a TM signal might be defined as an array of **4 x 16 bits**, where:

- **4** = TM factor (number of time slices)
- **16** = width (bits) of each ADC sample

Certain SciCompiler IP blocks already support TM signals, while others are in the process of being ported. In the interim, it is possible to exploit the **TM SPLITTER** and **TM PACKER** IP blocks to:

- **Split** a TM signal into multiple standard **bit vector** signals (one per time slice).
- **Process** each time slice in parallel (e.g., applying mathematical operations).
- **Repack** the individual bit vectors back into a TM signal for further processing.

This Application Note demonstrates how to use the TM SPLITTER and TM PACKER IP blocks to **rescale** the data by multiplying each sample by a constant **< 1**.

2 Signal Scaling algorithm

The example in the figure below shows a scaling factor of 0.3 applied on the input TM array

1. **TM SPLITTER Configuration**
    - In SciCompiler, instantiate the **TM SPLITTER** IP block.
    - Configure the input type as **TM** with a factor of **4** and a width of **16** bits (for the V2730).
    - This block creates four output ports, each a **16-bit** standard bit-vector signal in SciCompiler notation.
2. **Multiplier IP Setup**
    - Insert four multiplier IP blocks (or a single multiplier block used in a parallel manner if the tool allows).
    - Each multiplier takes a **16-bit** input from the splitter and produces a scaled 16-bit output.
    - The constant α (< 1) can be provided as a compile-time constant (like here) or from a register if dynamic reconfiguration is needed.

- Ensure you configure the internal fixed-point representation to accommodate fractional values without overflow or excessive truncation.

3. **TM PACKER Configuration**

   - Place a **TM PACKER** IP block at the outputs of the multipliers.
   - Set the TM factor to **4**, the bit width to **16**, and connect each multiplier output to the corresponding input of the TM PACKER.
   - This block regenerates a **4 x 16-bit** TM signal, called TM_OUT.

4.