

# Recognized Data Communication Patterns

Vector Fabrics™ tools are capable of detecting various data communication patterns based on behavior observed during dynamic analysis of a program for a given set of test data. The specific patterns known to the tools are detailed in this document. This list will be revised as new patterns are added to the capabilities. Any dependencies of unrecognized types will simply be reported as either a Compute Dependency or a Memory Dependency.

| Pattern Name         | Description   | Parallel Implementation  | Examples   |
|----------------------|---|--|--|
| <b>FIFO</b>          | <p>This is a <i>streaming</i> pattern. Each data element is written exactly once and read exactly once. Reading occurs in the same order as writing.</p> <p><u>Special properties:</u></p> <ul style="list-style-type: none"> <li>• <b>Syncs:</b> effectively the number of data elements transferred.</li> </ul>   | When the write and read operations are in separate tasks, a FIFO is indicated for communicating between the tasks.   | <p>The contents of an array are written in order and read in order using a loop. The read and write operations could occur in a single statement, such as</p> $a[i]=k*a[i-1]$  |
| <b>Windowed FIFO</b> | <p>This is a <i>streaming</i> pattern. Data elements are written to a range of addresses and read from a range of addresses. The ranges may be thought of as windows. The window used when writing may be different from the window used when reading. The window typically slides along the address range as data production and consumption progress.</p> <p><u>Special properties:</u></p> <ul style="list-style-type: none"> <li>• <b>Window size:</b> the maximum of the read and write ranges.</li> </ul> | When the write and read operations are in separate tasks, a typical implementation would be a FIFO of buffers. The window size correlates to the required buffer size. | A video filter needs to calculate a pixel based on the characteristics of surrounding pixels. The pixel array is written one pixel at a time, in order, but a small sub-array of pixels – say, 3x3 – is used to calculate the middle pixel. This sub-array, or window, slides along the row of pixels as each pixel is calculated in turn. |

## Recognized Data Communication Patterns

| Pattern Name              | Description  | Parallel Implementation   | Examples   |
|---------------------------|--|---|--|
|                           | <ul style="list-style-type: none"> <li><b>Syncs:</b> the number of buffer synchronizations required to complete the full transfer of data.</li> </ul>  |   |  |
| <b>Compute Dependency</b> | <p>The value of a variable to be stored in memory depends on a chain of calculations that starts with or includes another variable loaded from memory. Patterns cannot be detected with compute dependencies.</p> <p>vfAnalyst makes all variables in the chain of computation along the dependency visible.</p> | The computed values can usually stream from one task to the next.   | $x = *p$<br>...<br>$*q = f(x)$   |
| <b>Memory Dependency</b>  | <p>A value to be read (or consumed) from memory requires that the value already has been stored (or produced) earlier in the program. No specific pattern is detected.</p>   | The value must be communicated from the producing task to the consuming task, with appropriate synchronization. | $*p = f()$<br>...<br>$x = *q$<br>where $p$ and $q$ have the same value |