



A word size determination subset tool of **CADRE**

SynapC

1 Introduction

To reach, in a hardware implementation, the required performance, and to reduce power dissipation, word lengths have to be carefully determined, analyzed, and tested. This also means that if - as often is the case - the original algorithm was validated in floating point, a translation to fixed point is required.

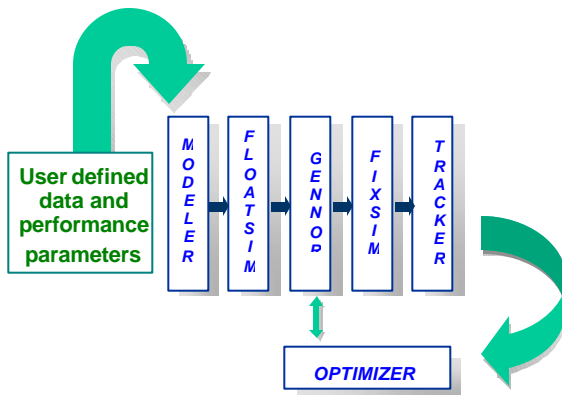
2 Overview

SynapC is a proprietary metric-based tool that **automatically determines the optimum number of bits** (word size) required to represent an algorithm operand in hardware. **SynapC** can process algorithms **with data dependencies**, and designers are not constrained with providing input range specifications. This means that designers can drastically reduce their development cycle by **jumping over time-consuming input** specifications development, floating to fixed point translations, and without concerns to jumps and loops associated with data dependencies.

SynapC, while handling several error models and designer generated specifications, automatically determines optimum size values with fewer. It **automatically searches for the best combination** that would meet the specified accuracy, thereby ensuring the algorithm stability.

With a word size optimized for specific performance, developers can generate hardware unique operands designs, thereby **reducing useless overhead, error buffers, or overestimated word size**. This produces more efficient streamlined designs with reduced hardware size, decreased costs, and lower power consumption.

3 The **SynapC** Matrix



SynapC accelerates hardware implementation developments and is integrated within a sub-set of CADRE's IDE.

Composed of five sub-process layers, **SynapC**'s architecture includes fixed and floating point simulators -**FixSim** and **FloatSim** respectively - that process data widths adjusted in one bit steps and are compatible with C, C++, SystemC, and ANSIC languages. The other layers include other sub-processes such as the **Gennor** that determines the operand format, the **Tracker** that manages and tracks various errors, the **Optimizer** that executes several optimization scenarios and identifies global system solutions. The user defines his input data, the algorithm to be analyzed, and performance parameters in a file called the **Modeler**.

4 **Synet**

One of **SynapC**'s key features is its built-in *networkability* in a multiprocessor or parallel environment through the **Synet** module. **SynapC** architecture is such that the **Synet** module can parallelize computation cycles by spreading them over an array of networked computers. **Synet** controls and supervises several **SynapC** executions, collects the results from each execution, and produces a final result.

**FOR FURTHER INFORMATION OR A WEB-BASED DEMO,
SEND YOUR REQUEST VIA OUR CONTACT US WEB PAGE**