Simultaneous Transmit And Receive (STAR) Messaging Protocol

By Earle Jennings, CTO, QSigma, Inc., earle.jennings@qisgmainc.com, (p) +1 510 292 8328

Keywords: In-chip communications network, inter-chip communications, MPI, many-core, fault resilience

Background

MPI, and the adaptation of various forms of Linux, enabled relatively inexpensive parallel processor systems. However, for HPC, in particular exascale systems, there are fundamental inefficiencies with the current approach to MPI. First, sending a message locks up a buffer until the message held in that buffer completes transmission. Second, receiving a message not only locks up a buffer to receive the message, but also locks it for as long as it takes the received data to be moved elsewhere, or processed in place. Third, routing a long message can lock up a router transfer point, stalling other messages from traversing that transfer point. Exascale systems add fault resilience challenges to communication protocols. These need to be addressed below the MPI layer to be fast enough that these systems do not stall. Also, data centers are plagued with malicious software attacks, often entering through their communication portals. These also need to be addressed below the MPI layer.

Introduction

This paper introduces a communications protocol, and its hardware implementations, resulting from QSigma's computer architecture experiments, which began in 2003. The objective of this effort included determining the required features for a communications protocol coupled to modules of cores inside chips that can seamlessly traverse a system of exascale, and above, data processing capacity. The overall system objective was to insure that the multipliers did not stall more than 10% of the time, and everything else in the system kept up. The initial test algorithms included Finite Impulse Response (FIR) filters, Fast Fourier Transforms (FFTs), and matrix inversion by Gaussian elimination. Initial experiments focused on single precision Floating Point (FP) implementations suitable for Digital Signal Processing (DSP) systems. We started with simulations, which were extremely labor intensive for the amount of results achieved. This led to a method of system level gedanken experiments. These involve a depiction, often a series of block diagrams, which are visualized to establish an initial estimate of the results. These are confirmed based upon systems analysis, hardware analysis, software technologies, and mathematics, as required by the experiment, and supported by available software tools. By 2009, a basic approach had emerged. By 2014, the test algorithms had been extended to include block LU decomposition, which dominates the performance requirements for High Performance Linpack (HPL). The computer and communications architecture were confirmed to deliver an exaflop for at least 8 hours, through a series of gedanken experiments. The experiments confirmed the multipliers did not stall more than 10% and everything else kept up in a proposed system with 256 cabinets, each containing 4K Data Processor Chip (DPC) stacks. Interestingly, with block LU decomposition, as with matrix inversion, the larger the system, the more efficient the multipliers became. This is due in no small part to this communications protocol. In 2016, it was retargeted from 20 Gbit/sec to 100 Gbit/sec transceivers.

The STAR Communications Network

The Simultaneous Transmit and Receive (STAR) message protocol transmits and receives a STAR message on every local clock cycle, except when responding to an uncorrectable error on reception. The response to such errors is automatic channel component replacement within, at most, a microsecond. Assuming a 1 ns clock, 200 Gbits/second can be delivered, and sent, on each STAR channel. The context of the message is interpreted at every STAR message core to determine its disposition and transfer. The context, and its interpretation, is under complete control of the program. This protocol removes all of the above inefficiencies for MPI in HPC and enables an exascale class computer.

Each STAR message contains a fixed length data package of a data payload of 128+5 bits, and a message routing context or direction of 32 bits. The STAR message includes an Error Correcting Code (ECC) of 35 bits. For example, the ECC may implement a Huffman coding scheme supporting 1 bit correction and 2 bit error detection for each 33 bits of the package.

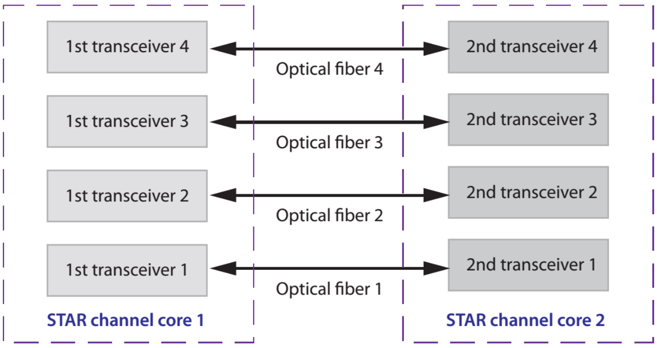
The data payload may be configured by the 5 bit code in many ways. Two example payloads are discussed, a first payload of two double precision numbers, each with two guard bits, and a second payload of one double precision number with two guard bits and a 64 bit integer index list. The first payload supports complex number arithmetic, as well as bulk downloading and uploading from the DPC chips, to save and restore the state of its cores. The second payload supports HPL pivot calculations, multi-grid operations and other sparse matrix operations.

The STAR message channels are further organized into a STAR bundle, containing data STAR channels and control and status STAR channels. For example, for the exascale system experiments, the STAR data channels include 16 instances of STAR message channels used for data transfers, and a spare instance used for fault resilience among these data channels. The control and status STAR channels include a task message channel, a transfer request message channel, and a spare instance used for fault resilience support of the control and status message channels.

A STAR communications network is a point-to-point network of nodes acting as sources and destinations, and routing nodes, called STAR Trinary Routers (STR). Each STR has three bidirectional STAR bundle links, which may be nodes or other STR instances. Each STR operates based upon the program that is executing. Each node can send and receive 16 data payloads every clock (1 ns) with the STR. Each node has a data bandwidth of 2 Terabits/sec in and out. Each node can be a module with a chip, or a chip. The STRs are implemented inside a chip as a communication module paired with a module(s) of cores as a component of the floor plan.

The STAR Protocol and Network Beyond a Single Chip

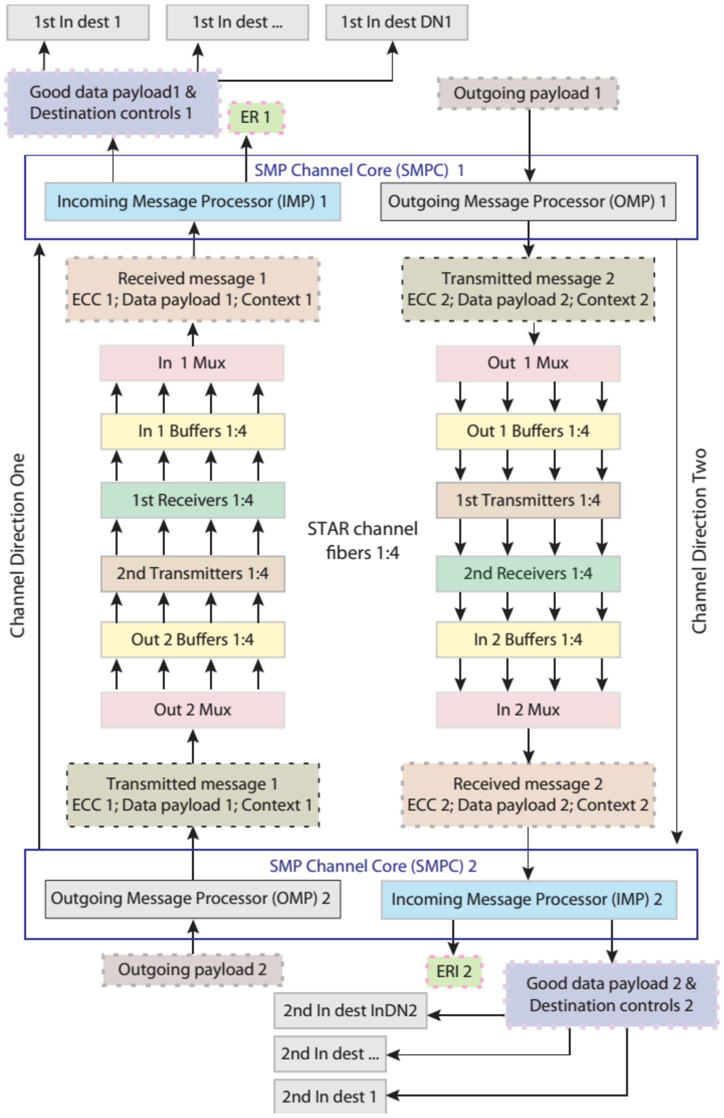
Communication between chips requires optical communications. Consider the individual messages communicated on a STAR message channel. This is shown in the following figure as an implementation consistent with today’s 100 Gbit/sec Ethernet opto-transceivers using four optical fibers, and associated transceivers, between two STAR channel cores.



Example of a multi-fiber STAR message channel.

The STAR channel cores, interfaced to a single STAR message channel, support four degrees of freedom in automated, local fault resilience in response to an uncorrectable message fault.

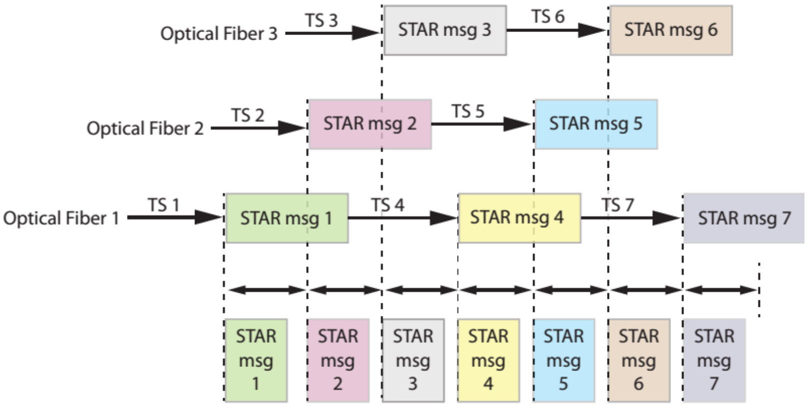
The opto-transceiver clocks can be slowed down. A faulty transmitter link to a receiver in one direction, can be replaced by the spare optical fiber in that direction. A faulty STAR message channel instance in one, or both, directions can be replaced with the spare STAR message channel. Two opto-fibers in a STAR message channel can be operated, rather than three, lowering the bandwidth to two STAR messages every three clock cycles.


1st In dest 1
1st In dest ...
1st In dest DN1
Good data payload1 & Destination controls 1
ER 1
Outgoing payload 1
SMP Channel Core (SMPC) 1
Incoming Message Processor (IMP) 1
Outgoing Message Processor (OMP) 1
Received message 1 ECC 1; Data payload 1; Context 1
Transmitted message 2 ECC 2; Data payload 2; Context 2
In 1 Mux
Out 1 Mux
In 1 Buffers 1:4
Out 1 Buffers 1:4
1st Receivers 1:4
1st Transmitters 1:4
STAR channel fibers 1:4
2nd Transmitters 1:4
2nd Receivers 1:4
Out 2 Buffers 1:4
In 2 Buffers 1:4
Out 2 Mux
In 2 Mux
Channel Direction One
Channel Direction Two
Transmitted message 1 ECC 1; Data payload 1; Context 1
Received message 2 ECC 2; Data payload 2; Context 2
SMP Channel Core (SMPC) 2
Outgoing Message Processor (OMP) 2
Incoming Message Processor (IMP) 2
Outgoing payload 2
ERI 2
Good data payload 2 & Destination controls 2
2nd In dest InDN2
2nd In dest ...
2nd In dest 1

Assume that two corresponding optical transceivers (1 and 2) can be locally clocked and operated at 130-140 GHz, with the STAR message channel operating at 100 Gbits/sec.

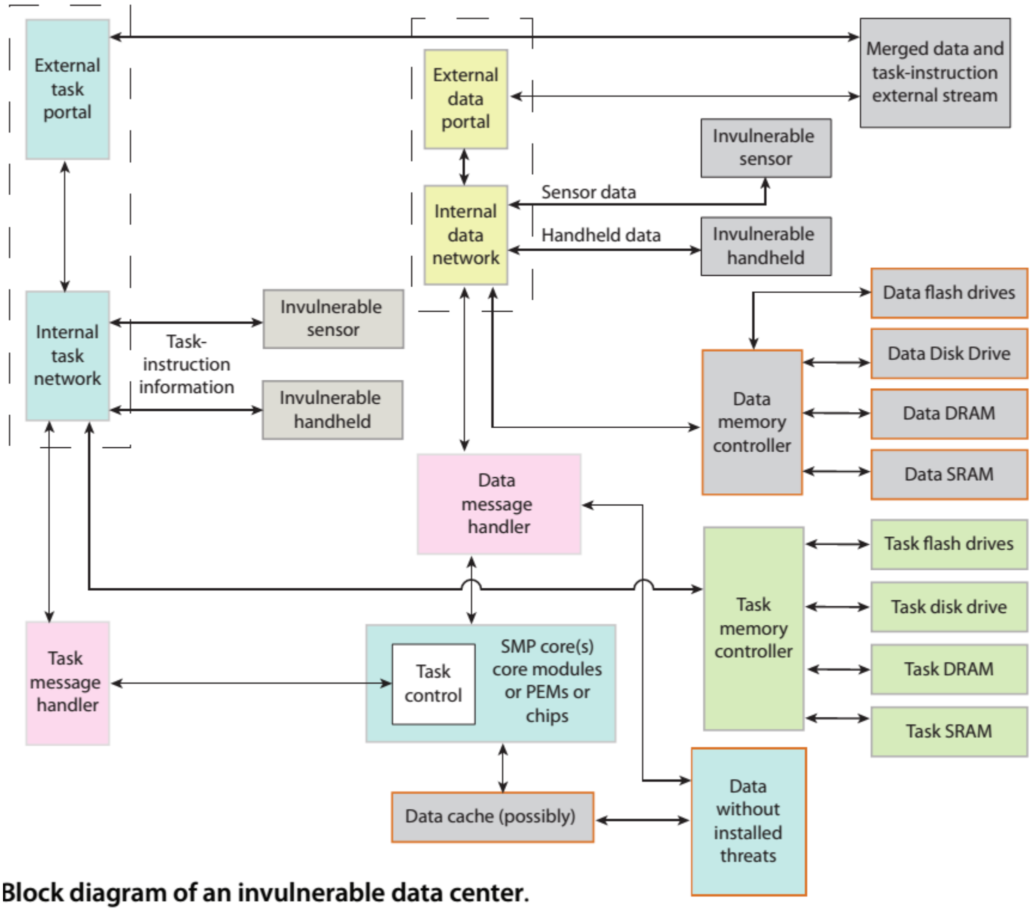
Only fibers 1 and 2 and only their transceivers need be active as shown.

Suppose no pair of corresponding transceiver can operate error-free at 100 Gbit/sec data transfer. This leads to transmitting STAR messages using three of the optical fibers.



TS k stands for the Training Sequence for the STAR msg k,
for k = 1 to 7.

Binary trees leave HPC system components vulnerable to bottlenecks. Extending the network to a Mostly Binary Tree, enables chip stacks, optical PCBs, etc., to have dual, or better, interfaces. For example, there are 30 to 33 Star bundles available at each cabinet to interface to its data center. Each optical channel is physically compatible with Ethernet. Each data STAR channel, of the available STAR bundles, can interface to multiple Ethernet networks. As a consequence, each cabinet can simultaneously communicate with 1K Ethernet networks, each with 100 Gbit/sec bandwidth. Today's data centers are vulnerable to many forms of malicious software attacks, for example viruses and rootkits. One common weakness is faulty access of data memory, leading to installed threats, which may then infect the various components of the data center, and may further infect other sites.



Block diagram of an invulnerable data center.

Tomorrow's invulnerable data center interfaces to less secure, general purpose, networks through a new interface. This new interface operates two primary portals to two separate internal STAR network components. One portal supports access to task management and program configuration, going to the task STAR channel of the various STAR links. A second portal supports data transfers to the STAR data channels. The invulnerable data center physically separates data memory, task-instruction memory, and their memory controllers. There are no transfer paths from one form of memory to the other. No data-related operation can alter a task, or an instruction, residing in the task-instruction memory. This removes the opportunity for viruses and rootkits to infect cores, DPCs, handhelds and networked sensors.