Ganglia Mark 1: A Dynamic Message Routing Surface

This paper describes a means for exchanging message transactions between nodes that do not share a common COM port using message frames whose headers hold source routing information that is updated incrementally as the frame moves between contiguous nodes that have been programmed to serve as part of a surface, or fabric, for this purpose. In the polyFORTH system, for example, all otherwise fallow nodes are programmed in this way and are utilized for most communications between the polyFORTH virtual machine and other interfaces elsewhere on the chip.

Each transaction with a target node consists of delivering to one of its ports a focusing call instruction word followed by one or more words of arbitrary payload, and receiving one or more words of arbitrary reply. Transactions begin by feeding a Ganglion frame from a source node outside the surface into the first Ganglion node from which it propagates along a path defined in that frame's header. In this paper the Snorkel is used as an example of a source node.

The Mark 1 protocol is simple, brief, and requires only 32 words of resident code in each node that is used to route messages. Its routing description is sufficient for straightforward traffic in a chip that has a large, uncongested routing surface. The Mark 2 protocol should be used when the simple, three-legged routing is insufficient.

Related Documents

AN003: SRAM Control Cluster Mark 1
AN010: Snorkel Mark 1
DB006: polyFORTH Supplement for G144A12
Many additional applications such as AN001, AN008 and AN012 utilize this mechanism.

Contents

1. Problem Statement

Within a node cluster static data routing is the fastest and simplest way for nodes to communicate with one another. Between clusters, though, there are times when dynamic message routing is needed. For example, in many cases functions cluster around relevant I/O pins located at the edges of the chip. Data flow may be episodic rather than continuous. In such cases generalized traffic control at path crossings can be expensive of energy and code. The compelling motivation in this case was the need for high-level polyFORTH software to communicate with any relevant I/O nodes, such as SPI flash, using dynamic routing controlled and defined directly by the high-level software and tailored for the application.

2. Our Solution

Chuck Moore was using an experimental method of dynamic routing based upon a source-routed message frame. The method depended on pre-programming all otherwise unused nodes with a program to interpret and process those frames incrementally, resulting in a communications fabric he called *ether*. We adapted the program and the message frame definition to suit the requirements of this application, naming the result a *surface of Ganglia*. Each node programmed for this purpose is called a *Ganglion*, the message format a *Ganglion frame*, and the act of performing the actions prescribed by such a frame is called a *Ganglion transaction*.

Each Ganglion Transaction consists of delivering one or more words of *payload* from a *source node* to a *target node* and receiving one or more words of *reply*, using a rectilinear routing path with a maximum of three segments. When the payload is delivered to the target node it is preceded automatically by a focusing call to the port through which the rest of the payload is delivered and the reply received. No assumptions are made about software in the target node; even the focusing call does not necessarily imply that the target is executing the port being used. The Ganglion frame is structured as follows, with a 5-word header and variable length payload/reply:

<table>
<tr><th colspan="6">18-bit word</th><th>Function</th></tr>
<tr><td colspan="6">Focusing call (updated for each port crossing)</td><td>First word of header, also included in payload delivery</td></tr>
<tr><td colspan="6">call to pump routine in ganglia</td><td rowspan="5">Remaining header (four words), not delivered to target
- Focusing call (above) and path updated at each step
- Header stripped when delivering payload at destination
- Direction codes (d_x) are 0,1,2,3 are cardinal E,W,N,S.
- Distance (n_x) 1-relative. 0 means deliver in direction d_x if all remaining segments are null.</td></tr>
<tr><td colspan="6">Path remaining. $n_x|d_x$ defines segment x. A segment numerically all zero is null.</td></tr>
<tr><td>n3</td><td>d3</td><td>n2</td><td>d2</td><td>n1</td><td>d1</td></tr>
<tr><td colspan="6">reply count, Y-1</td></tr>
<tr><td colspan="6">payload count, X-1</td></tr>
<tr><td colspan="6">Payload (X words)</td><td>Sent with outbound frame and delivered after focusing call at destination</td></tr>
<tr><td colspan="6">Reply (Y words)</td><td>Transferred from destination back to originator</td></tr>
</table>

A Ganglion transaction propagates outward from the Snorkel through a sequence of Ganglia along its path. Transactions are driven by the Snorkel Mark 1 between external SRAM and any port on the Snorkel node that connects to the surface of Ganglia, although nodes running other software may drive transactions if this creates no irresolvable routing problems. Each Ganglion interprets and updates the header to determine what node is next. That node is passed the focusing call and, if it is not the target node, the updated header follows. This is followed by the payload, after which the Ganglion waits to receive the reply, transferring it backward along the path. When each Ganglion has finished passing the reply, that node normally returns to **warm** so it's ready for any port executable stimulus, including another Ganglion transaction.

The path segments are interpreted right to left, as taking **n** steps in cardinal direction **d**. Steps are numbered from the first node seeing the message, typically the first node outside a snorkel. When all segments have been processed such that no nonzero **n** are left, delivery occurs to the next node in the most recent direction. A segment with initial **n** zero denotes a change of direction for delivery without taking a step; such a segment may not be to the "East" (**d**=0.)

3. Implementation

The program loaded into the upper half of RAM in each Ganglion is generated using the following template after a table has been generated at location x20 giving call instructions for the ports leading in the cardinal directions used by the encoded routing. This table is used to compensate for the pattern of node orientations.

```
ganglia route messages much like chuck's 'sea'
delivers exchanges of 1 or more word out and,
1 or more word reply to arbitrary nodes with,
source routing. message structure is...,
,
..focus call to port always there,
....pump call interganglion only,
....path see below inter only,
....reply count words-1 inter only,
....payload count words-1 inter only,
..payload always there,
..reply always there,
,
path has 3 6-bit runs low order taken first,
run encoded nnnndd path has 3 runs,,
...2-bit direction rlud 0123,
...4-bit count 1-relative zero deliver immed,
......3rd run must be lt 8,
example from 708 to 617 go down 1 and right 8,
...8 0 1 3 packed as 807 in octal.,
,
when an exchange is finished all,
ganglia are back where they were,
on receipt of focusing call.
```

```
406 list
- ganglion template,
,
a whence b whither,
msg is focus call path cin-1 cout-1 code,
,
20 org rlud r--- --l- ---u -d--
aim 24 p-pa dup 3 and 20 or b! @b dup b! !b,
@p ; / pump leap
whither 2A p -4 . + aim !b path !b,
cnts @ dup push !b @ dup push !b
payload 30 begin @ !b unext,
...begin @b ! unext ;
/8 32 n-n 2/ 2/ 2/ ;
pump 33 then pop a! @ dup
turn pp 3C and if drop whither ; then,
...drop dup FC0 and if drop /8 /8 dup turn ;,
then drop aim cnts @ push @ push payload ;,
40
```

Path encoding is interpreted right to left, starting with d_1 specifying cardinal direction (0 right or East, 1 left or West, 2 up or North, 3 down or South) and n_1 specifying a number of steps to be taken in that direction. Counting begins from the first Ganglion, not from the Snorkel. If a second leg is needed, it is encoded in d2/n2 and a third is in d3/n3. Unused legs must be in the high order position(s) and are all zero; this precludes a zero length East segment in any position but the first.

The path is interpreted in each Ganglion, selecting a port based on cardinal directions and either passing a Ganglion frame to the next Ganglion or delivering a payload to the target node when the path has been exhausted. The target node is given a focusing call at the beginning, in case the payload is port executable; whether the payload is code or data depends on what the target has been programmed to expect. Reply is passed backward through the port from which the message was received, after which the ganglion returns to whatever address received the initial focusing call. Normally Ganglia start at warm and so normally this return will be to the node's idle multiport execution address.

Each Ganglion is completely devoted to the processing of a message from the time it receives the focusing call at the start of a message until it has successfully written the last word of reply through the port from which it received the frame. The chain of Ganglia actively processing a transaction is an impenetrable wall while this is going on. Although it is possible for multiple paths such as ganglion messages and/or IDE wires to be active in a chip concurrently, only one of them may pass through a given node at a given time and the normal mechanism in a Ganglion cannot resolve multiple incoming focusing calls that are capable of occurring within an instruction time of each other. When multiple paths cross only once and this timing constraint is observed, traffic control is automatic. When such paths cross more than once, deadlock becomes possible so system design must ensure that does not happen.

Nothing precludes building special "plumbing fixtures" to facilitate traffic control. A node may be programmed to resolve effectively simultaneous arrival of focusing calls by polling io and selecting which port to process. It's also possible to program a node for acting as a transparent bridge with predefined output port for each input port, thus permitting two active paths of any sort to cross. The cost of such polling, in the F18A, is the power bill for a node running at 100% duty cycle.

4. Usage Examples

All examples shown here use polyFORTH, but there is nothing to preclude your programming F18 nodes to drive their own transactions across a surface of Ganglia so long as path crossing rules are respected.

4.1 polyFORTH Mechanism

The outgoing part of a Ganglion frame is normally built in external SRAM as a sequence of double words so it can be transmitted as 18-bit data using the Snorkel. Block 28, below, defines the essential vocabulary, and block 31 gives a simple example of using this vocabulary.

```
2428
 0 Tools for extending the memory mastering (Snorkel/Ganglia) I/O
 1    that's in the nucleus.
 2
 3 :UP  and  :RIGHT  are port adrs for starting channel programs.
 4
 5 'reload  is a chain of executable code that returns when done.
 6 +RELOAD  adds code to the reload chain.  Usage:
 7    HERE ] <stuff to do before existing chain>  +RELOAD [
 8
 9
10
11
12
13
14
15
```

```
28
 0 ( Snorkel/Ganglia)   HEX
 1 HEX  2145 1 2CONSTANT :UP  21D5 1 2CONSTANT :RIGHT
 2      2115 1 2CONSTANT :DOWN  2175 1 2CONSTANT :LEFT
 3
 4 : 2, ( d)   , , ;   : W, ( n)   0 2, ;
 5 : run   CONSTANT DOES  4 * R> @ OR ;
 6    0 run rr   1 run ll   2 run uu   3 run dd
 7 : ,path ( n n n)   40 * OR  40 U* , OR , ;   DECIMAL
 8
 9 : +RELOAD ( a)   'reload @ \ AGAIN  'reload ! ;  IMMEDIATE
10
11 : DRIVE ( n)   DUP 20 < IF 2400  ELSE 480
12       THEN ['] SHADOWS 1+ !  DRIVE ;
13 ( Reset it)  OFFSET @ 1200 / DRIVE
14
15
```

```
2431
 0 Timing support using free running clock node accessed via the
 1    snorkel.  This version accesses node 516 which is monitoring
 2    the 6 MHz clock output by FTDI chip (typically port B).
 3 tfre  is a 32-bit free running clock with resolution 166 ns and
 4    wrapping at 715.827 seconds.
 5 tCLK  is a 32-bit free running clock with resolution 10.9926 ms
 6    and wrapping every 535 days after boot.
 7 Csnk  facil var protecting chan pgm & ganglia msg.
 8
 9 COUNTER  returns 10 MHz counter, takes about 46.7 us.
10 TIMER  displays time in usec since the time given.
11 MS  delays about n milliseconds.
12
13 @CLK  updates both free running clocks.  Must be executed more
14    often than the high res clock wraps to work properly.
15
```

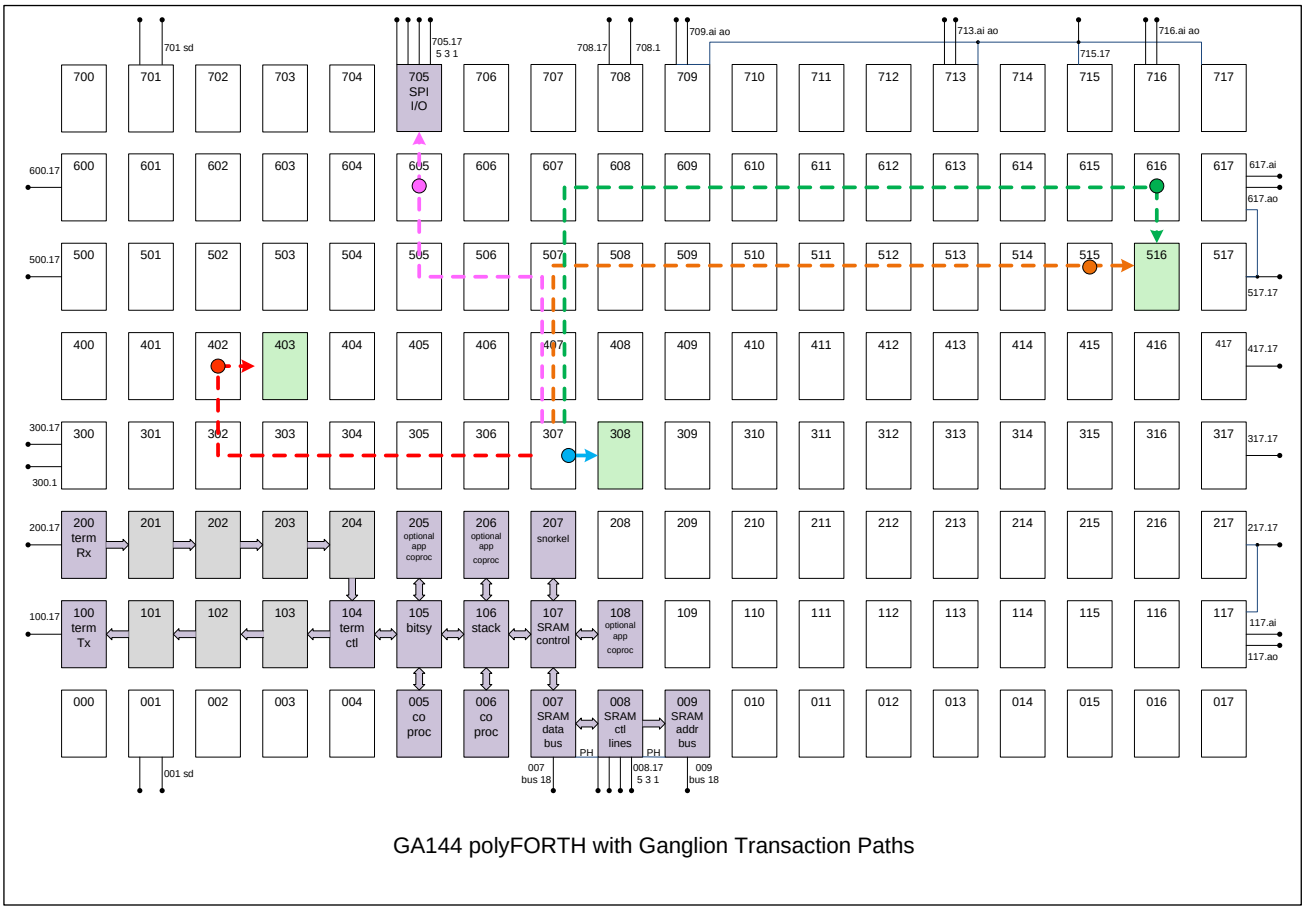
```
31
 0 ( 6MHz clock)   2VARIABLE tfre  2VARIABLE tCLK  HEX
 1    VARIABLE Csnk
 2 HERE  :DOWN 2,  1 , 2033 ,  2 uu 8 rr 0 ,path  ( rsp) 1 W,
 3    ( pay) 0 W,  1 , 2200 ,    ( lng) HERE OVER - 2/ 1-
 4 CREATE ?CK   :DOWN 2,   o18 ,  ( lng) W,  ( ?ck) DUP W, 0B +
 5    i16 ,  1 W,  tfre W,   HERE  FIN ,  CONSTANT CKfin
 6 : COUNTER ( - d)   2200  BEGIN [ SWAP ]  Csnk GRAB
 7    LITERAL !  ?CK CKfin +SNORK sDONE  tfre 2@  Csnk RELEASE ;
 8 : -tmr   220A AGAIN ; RECOVER  -tmr 0 tCLK 2!  DROP  DECIMAL
 9
10 : TIMER ( d)   COUNTER D- DNEGATE  10 6 M*/
11    <# # 46 HOLD #S #> TYPE SPACE ;
12 : MS ( n)   6000 U*  COUNTER D+  BEGIN PAUSE
13       2DUP COUNTER ( CR 2OVER D. 2DUP D.) D< UNTIL  2DROP ;
14 : @CLK ( - d)  COUNTER DUP tCLK 1+ @!  OVER XOR 0< IF
15       DUP INVERT 0< IF  1 tCLK +!  THEN THEN 2DROP  tCLK 2@ ;
```

The following words facilitate compilation of Ganglion frames:

`:UP  :DOWN  :RIGHT  :LEFT` Double precision values of port focusing call instructions
`2, (d)` Compiles a double precision value into memory for 18-bit snorkel transfer
`W, (n)` Compiles a single precision value into memory for 18-bit snorkel transfer
`rr ll uu dd` all **`(n-s)`** Used to build path segments
`,path (s1 s2 s3)` Compiles an 18-bit path description into memory for the three segments given. Each unused segment is written as zero.

The example in block 31 compiles a 12-word structure in polyFORTH memory that will generate a Ganglion frame of six words as follows:

`:DOWN 2,` Compiles a focusing call for the down port of the first Ganglion, node 307, next to the snorkel.
`1 , 2033 ,` Compiles a `call` instruction to `pump` in the Ganglion.
`2 uu 8 rr 0 ,path` Compiles a path that goes upward 2 nodes from 307, to 507, then goes rightward eight nodes to node 515, after which payload is delivered rightward to target node 516 (see next page for more information on path definitions.)
`1 W,` Compiles reply count, meaning a 2-word reply is expected.
`0 W,` Compiles payload count, meaning a 1-word payload follows.
`1 , 2200 ,` Compiles the payload: A call to location 0 in the target node with P9 set for Extended Arithmetic Mode. Thus the target node receives two words, the appropriate focusing call followed by this call to zero, and will return two words of reply through the same port.



GA144 polyFORTH with Ganglion Transaction Paths

In the block diagram above, all white nodes have been loaded with Ganglion code and are suspended at their appropriate multiport execution addresses. The dashed lines show five Ganglion paths discussed here. In each case the Snorkel is directed to use its **down** port and Ganglion frames begin with **down** to focus node 307 on the Snorkel. In each case the path leads to the last Ganglion, denoted by colored circles, for delivery to the target node (arrow). The examples are color coded to match the graphical paths above.

`2 uu 2 ll 1 uu ,path` This path is used to communicate with code in node 705 that controls SPI operations such as the flash memory on the Evaluation Board. Delivery is in same direction as the last segment in the path (upward).

`2 uu 8 rr 0 ,path` This is the path used in the example above from block 31 for communicating with the clock monitoring code in node 516. Note that the third segment is not used.

`3 uu 9 rr 0 dd ,path` This is an alternative path for communicating with node 516. In this example, the third segment is of zero length, causing the last Ganglion (node 616) to deliver downward. If this segment were written as zero, the delivery would be made rightwards to node 617.

`0 rr 0 0 ,path` When the first segment is of zero length, it simply causes the first (and last) Ganglion to deliver in the given direction. *This is the only case in which a zero length rightwards path may be used.*

`5 ll 1 uu 0 rr ,path` ***This path will not work.*** `0 rr` is equivalent to zero and will not be processed because it is not the first segment. In this case, delivery would be made to node 502 instead of 403. (Delivery to the left, using `0 ll` , would work as expected.)

GreenArrays®

Application Note AN011 *Revised 6/08/13*

IMPORTANT NOTICE

GreenArrays Incorporated (GAI) reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to GAI's terms and conditions of sale supplied at the time of order acknowledgment.

GAI disclaims any express or implied warranty relating to the sale and/or use of GAI products, including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright, or other intellectual property right.

GAI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using GAI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

GAI does not warrant or represent that any license, either express or implied, is granted under any GAI patent right, copyright, mask work right, or other GAI intellectual property right relating to any combination, machine, or process in which GAI products or services are used. Information published by GAI regarding third-party products or services does not constitute a license from GAI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from GAI under the patents or other intellectual property of GAI.

Reproduction of GAI information in GAI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. GAI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of GAI products or services with statements different from or beyond the parameters stated by GAI for that product or service voids all express and any implied warranties for the associated GAI product or service and is an unfair and deceptive business practice. GAI is not responsible or liable for any such statements.

GAI products are not authorized for use in safety-critical applications (such as life support) where a failure of the GAI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of GAI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by GAI. Further, Buyers must fully indemnify GAI and its representatives against any damages arising out of the use of GAI products in such safety-critical applications.

GAI products are neither designed nor intended for use in military/aerospace applications or environments unless the GAI products are specifically designated by GAI as military-grade or "enhanced plastic." Only products designated by GAI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of GAI products which GAI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

GAI products are neither designed nor intended for use in automotive applications or environments unless the specific GAI products are designated by GAI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, GAI will not be responsible for any failure to meet such requirements.

For current information on GreenArrays products and application solutions, see www.GreenArrayChips.com

Mailing Address: GreenArrays, Inc., 774 Mays Blvd #10 PMB 320, Incline Village, Nevada 89451
Printed in the United States of America
Phone (775) 298-4748 fax (775) 548-8547 email Sales@GreenArrayChips.com