Ganglia Mark 2: 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 2 protocol supports any necessary routing path and is suitable for debugging and other ad hoc communications in single or multi-chip environments, yet still fits in the upper half of node RAM. This version is incorporated in and used by polyFORTH starting with level 02c. In addition the various caveats of Mark 1 have been eliminated and the messages are easier to generate automatically.

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. A second purpose is to support richer routing capability in the second-generation IDE.

2. Our Solution

This is a method of dynamic routing based upon a source-routed message frame. It depends on pre-programming all otherwise unused nodes called *Ganglia* with a program to interpret and process those frames incrementally, resulting in a communications fabric we call a *surface of Ganglia*. The message format is called a *Ganglion frame*, and the act of performing the actions prescribed by such a frame is a *Ganglion transaction*. The Mark 2 alleviates Mark 1 limitations and caveats while enabling new capabilities such as multi-chip IDE operations.

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 as many segments as the circumstances require. 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 header of 5 or more words and variable length payload/reply:

18-bit word	Function
Focusing call (updated for each port crossing)	First word of header, also included in payload delivery
call to **`pump`** routine in ganglia	Remaining header (≥4 words), not delivered to target - Focusing call (above) and path updated at each step - Header stripped when delivering payload at destination - Direction codes (**d**) are 0,1,2,3 are cardinal E,W,N,S. - Distance (**n**) 1-relative. Nonzero means decrement (**n**) and take a step in direction (**d**). 0 means deliver in direction (**d**) if and only if (**e**) set marking this as the last segment; otherwise the segment is deleted and the next processed.
reply count, Y-1	
payload count, X-1	
One or more words of path remaining. Each word defines a segment. **`e- ---- nnnn nnnn nndd`**	
Payload (X words)	Sent with outbound frame and delivered after focusing call at destination
Reply (Y words)	Transferred from destination back to originator

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 returns to **`warm`** (or whatever other execution address it may have been given before the transaction began) so it's ready for any port executable stimulus, including another Ganglion transaction.

The path segments are interpreted sequentially, 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, delivery occurs to the next node in the most recent direction. A segment with initial **n** zero denotes a change of direction (for delivery if **e** set) without taking a step. Zero segment without **e** is effectively a no-op.

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.

```
4691
Background nodes are filled by default with ganglia; routing
   code is in x20..x3F thus they may support other messaging
   methods such as IDE or "neural" as well until node has been
   programmed in some other way.

Routing directions are geographic East-West-North-South.  Each
   node has a table "ewns" that translates these directionss
   into port addresses.

Four bins hold node orientations:
   1716: oee 100    1717: ooo 101
   1714: eee 000    1715: eoo 001

This block assembles the four of these bins, differing only
   in their ewns tables.

```

```
2291
( 1714-7 Mk2 Ganglia)

( Nodes oee)  1716 100  OVER .  ASM[  NODE BIN  ERS
# x20 org  : ewns  r--- --l- ---u -d--  # 1 FH LOAD  >BIN ]ASM

( Nodes ooo)  1717 101  OVER .  ASM[  NODE BIN  ERS
# x20 org  : ewns  --l- r--- ---u -d--  # 1 FH LOAD  >BIN ]ASM

( Nodes eee)  1714 000  OVER .  ASM[  NODE BIN  ERS
# x20 org  : ewns  r--- --l- -d-- ---u  # 1 FH LOAD  >BIN ]ASM

( Nodes eoo)  1715 001  OVER .  ASM[  NODE BIN  ERS
# x20 org  : ewns  --l- r--- -d-- ---u  # 1 FH LOAD  >BIN ]ASM



```

```
4692
The improved Mark 2 Ganglia deliver exchanges of 1 or more pay-
   load words and 1 our more reply words to any arbitrary node
   with more flexible, consistent source routing than did Mk1.
Msg: focus call (to port)        (always there)
       pump call                  (interganglion only)
       rc reply count, words-1    (interganglion only)
       pc payload count, words-1  (interganglion only)
       path <varies, see below>   (interganglion only)
     payload (rc+1)             (always there)
     reply (rc+1)               (always there)
Path:  Sequence of >=1 word with one run in each.  Coding is:
         e- --- nnnn nnnn nndd
       d is cardinal direction 0123 EWNS
       n is step count 1-relative; zero is deliver immediate.
       e marks last run.
When an exchange finishes all ganglia are as they were initially
```

```
2292
   ( MK2 Ganglia)   ASM[

: aim ( pth - pth a)  dup 3. and  # ewns S>D lit xor b!
   @b dup b!  ( focus) !b  @p ;  ( pump) leap

: whither ( pth)  -4. . + aim  ( pump call ) !b
   ( cnts)  r> !b r> !b ..
   ( path)  ahead begin  drop @  SWAP  then  dup !b -until
: payload  begin @ !b unext  begin @b ! unext ;

: pump  then  r> a!  ( cnts) @ dup >r  @ dup >r  >r >r
: turn  begin  @ dup xFFC. and if  ( next leg) drop whither ;
   then drop -until  ( deliver) aim r> r> payload ;



```

Or, the second block only as represented in colorForth:

```
improved ganglia deliver exchanges of 1 or,
more words out and 1 or more word reply to any
arbitrary nodes with more flexible source,
routing than did mark 1 and no gotchas.,
message structure...,
,
..focus call to port always there,
....pump call interganglion only,
....reply count words-1 inter only,
....payload count words-1 inter only,
....path see below inter only,
..payload always there,
..reply always there,
,
path is a sequence of 1 or more words with one
..run in each, encoded as follows...,
....e- ---- nnnn nnnn nndd,
..d is cardinal direction ewns 0123,
..n is count 1-relative zero deliver immed,
..e marks last run,
,
when an exchange is finished all,
ganglia are back where they were,
on receipt of focusing call.
```

```
414 list
- ganglion mark 2,
,
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,
cnts 2E/1 pop !b pop !b,
path 2F/2 .. ahead begin drop @ swap then,
..dup !b -until
payload 32 begin @ !b unext,
...begin @b ! unext ;
pump 34 then pop a!,
cnts @ dup push @ dup push push push,
turn 37 begin @ dup FFC and if,
next leg drop whither ; then -until,
deliver drop aim pop drop pop drop payload ;,
3F
```

Path encoding is interpreted sequentially from the first word transmitted, with **d** specifying cardinal direction (0 right or East, 1 left or West, 2 up or North, 3 down or South) and **n** specifying a number of steps to be taken in that direction. Counting begins from the first Ganglion, not from the Snorkel. If a second segment is needed, it is encoded in the next word and so on. Unused segments are not included in the message, and the final segment is marked with its sign bit.

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
Tools for extending the memory mastering (Snorkel/Ganglia) I/O
   that's in the nucleus.

:UP  and  :RIGHT  are port adrs for starting channel programs.

+N +S +E +W  compile a path segment for turning in the desired
   cardinal direction and taking  n  steps
deliv  marks the last segment to deliver to the next node in the
   direction in which it turned.

'reload  is a chain of executable code that returns when done.
+RELOAD  adds code to the reload chain.  Usage:
   HERE ] <stuff to do before existing chain>  +RELOAD [
```

```
28
( Snorkel/Ganglia MK2)   HEX
HEX  2145 1 2CONSTANT :UP  21D5 1 2CONSTANT :RIGHT
     2115 1 2CONSTANT :DOWN  2175 1 2CONSTANT :LEFT

: 2, ( d)   , , ;   : W, ( n)   0 2, ;
: run   CONSTANT DOES ( n)   4 * R> @ OR W, ;
   0 run +E   1 run +W   2 run +N   3 run +S
: deliv   2 HERE 2 - +! ;   DECIMAL

: +RELOAD ( a)   'reload @ \ AGAIN  'reload ! ;  IMMEDIATE

: DRIVE ( n)   DUP 20 < IF 2400  ELSE 480
      THEN ['] SHADOWS 1+ !  DRIVE ;
( Reset it)  OFFSET @ 1200 / DRIVE
```

```
2431
Timing support using free running clock node accessed via the
   snorkel.  This version accesses node 516 which is monitoring
   the 6 MHz clock output by FTDI chip (typically port B).
   For example connect J13 pin 1 to J21 pin 3 (517.17).
tfre  is a 32-bit free running clock with resolution 166 ns and
   wrapping at 715.827 seconds.
tCLK  is a 32-bit free running clock with resolution 10.9926 ms
   and wrapping every 535 days after boot.
Csnk  facil var protecting chan pgm & ganglia msg.

COUNTER  returns 6 MHz counter, takes about 46.7 us.
TIMER  displays time in usec since the time given.
MS  delays about n milliseconds.

@CLK  updates both free running clocks.  Must be executed more
   often than the high res clock wraps to work properly.
```

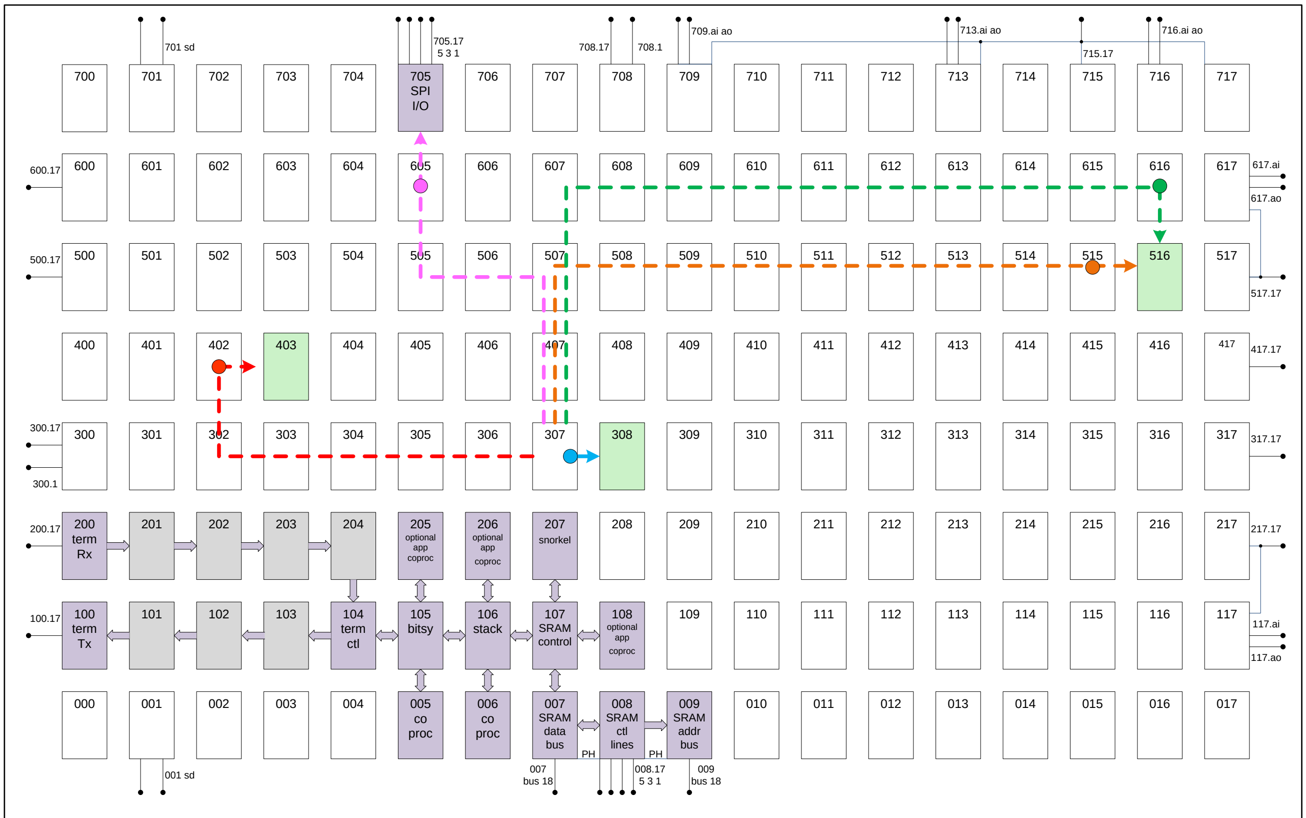
```
31
( 6MHz clock)   2VARIABLE tfre  2VARIABLE tCLK  HEX
   VARIABLE Csnk
HERE  :DOWN 2,  1 , 2034 ,  ( i/o) 1 W, 0 W,  2 +N 8 +E deliv
   ( pay)  1 , HERE SWAP 2200 ,   ( lng) HERE OVER - 2/ 1-
CREATE ?CK   :DOWN 2,   o18 ,  ( lng) W,  ( ?ck) W,
   i16 ,  1 W,  tfre W,   HERE  FIN ,  CONSTANT CKfin
: COUNTER ( - d)   2200  BEGIN [ SWAP ]  Csnk GRAB  LITERAL !
   ?CK CKfin +SNORK sDONE  tfre 2@  Csnk RELEASE ;
: -tmr   220A AGAIN ; RECOVER  -tmr 0 tCLK 2!  DROP  DECIMAL

: TIMER ( d)   COUNTER D- DNEGATE  10 6 M*/
   <# # 46 HOLD #S #> TYPE SPACE ;
: MS ( n)   6000 U*  COUNTER D+  BEGIN PAUSE
      2DUP COUNTER ( CR 2OVER D. 2DUP D.) D< UNTIL  2DROP ;
: @CLK ( - d)  COUNTER DUP tCLK 1+ @!  OVER XOR 0< IF
      DUP INVERT 0< IF  1 tCLK +!  THEN THEN 2DROP  tCLK 2@ ;
```

The following words facilitate compilation of Ganglion frames with static paths:

- **`: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
- **`+E +W +N +S`** all **`(n)`** Compiles a path segment to turn in the desired direction and step **`n`** nodes
- **`deliv`** Marks the last segment to deliver to the next node in the direction to which it turned.

The example in block 31 compiles a 14-word structure in polyFORTH memory that will generate a Ganglion frame of seven words as follows:

- **`:DOWN 2,`** Compiles a focusing call for the down port of the first Ganglion, node 307, next to the snorkel.
- **`1 , 2034 ,`** Compiles a `call` instruction to pump in the Ganglion.
- **`1 W,`** Compiles reply count, meaning a 2-word reply is expected.
- **`0 W,`** Compiles payload count, meaning a 1-word payload follows.
- **`2 +N 8 +E deliv`** 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 , 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 +N 2 +W 1 +N deliv` 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 +N 8 +E deliv` 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 +N 9 +E 0 +S deliv` 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 +E deliv` When the first segment is of zero length, it simply causes the first (and last) Ganglion to deliver in the given direction.

`5 +W 1 +N 0 +E deliv` ***This path was impossible in Mark 1.*** Delivers to node 403 through its **right** port as expeceted.

`1 +N 7 +W 1 +S 1 +N 8 +E 0 +N deliv` ***This path was impossible in Mark 1.*** The first two segments reach node 400. The third, 1 +S steps across the port bridge into node 10400 in the target chip if using the bridge and wired as on the evaluation board. The fourth steps up to row 10500, the next segment goes to node 10508, and we deliver upward from there into the target node (10608).

IMPORTANT NOTICE

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

Mailing Address: GreenArrays, Inc., 821 East 17th Street, Cheyenne, Wyoming 82001

Printed in the United States of America

Phone (775) 298-4748 fax (775) 548-8547 email Sales@GreenArrayChips.com