Transparent Port Bridge

Emulating a COM Port Connecting Two Chips

Early on, we developed a method of connecting two chips by their synchronous boot pins 300.1 and 300.17 using dedicated code in node 300 of both chips. The goal was to make it appear to node 400 of each chip that its **UP** port connected directly to the **UP** port of node 10400 (for second chip in extended node numbering notation). This allowed our basic tools such as boot streams, the IDE and the Ganglia to traverse the gap between two chips without having to change anything in the low level code. Everyone who has used the word `autotest` on an Evaluation Board has employed this tool. The key difference between this mechanism and a real port is that **IO** in node 400 can't be used to learn if the port is ready for us to write. This version waits for an acknowledgment from the other chip's node 300, signifying that the other chip has written the data to its **UP** port, before considering a read of its own **UP** port again. This implements full flow control which, if one does not read **IO**, behaves identically with a low speed COM port. All other rules governing COM ports apply.

When running, the following code resides in both 300 and 10300:

```
Transparent bridge between two chips' node 400 UP ports using
   nodes 300 for the connection.  18-bit data move MSB first
   on pin 1 on falling clock edges, flow ctl by ack of readiness
   for more: receiver drives clk hi after disposing word.

?lo  spins till clock line is low, stack destroyed.
dly  waits about 88ns and writes a new value to io.
1bt  given io value, sends bit with clk hi then drives clk lo.
wpd  sets weak pull-down on both lines; zro delays first.
18o  transmits an 18-bit word, MSB first, lines left in WPD.
18i  rcvs a word by setting lines hi-z, reading bits on falling
   clk edges.  Delivers to node 400 and acks by spiking clk hi.
idl  waits for a word from up (400) or a high clock on line then
   moves one word, with flow control, in proper direction.
The RAM load includes a boot frame header at 05 preceding the
   actual code at 8.  The bias of 3 preserves address alignment.
```

```
1904 300 ( Sync Bridge w/flow)   OVER . ASM[  NODE BIN ERS
# 08 org  : orgn  : ?lo  begin @b inv -until ;
: dly ( b)  40. for unext !b ;  ( ~88 ns)
: 1bt ( b)  dup dly  x10000. xor dly ;
: zro  x10001. dly ;   : wpd  x10001. !b ;
: 18o ( w - 0)   17. for begin
   -while x30003. 1bt 2* next  drop wpd ;
   then   x30002. 1bt 2* next  drop wpd ;
: 18i  ( HiZ) dup xor !b  17. for  begin @b -until
     begin @b inv -until  inv 2. and 2/  a 2* xor a!  next
   a up a! !  ( ack) x30001. 1bt wpd
: idl  # --lu alit a! ..  @ @b -if  ( wire) drop 18i ;
   then  ( port) drop 18o  begin @b -until  ?lo idl ;
: ent   io b!  wpd  ?lo idl ;   : last
here # 8 FORTH -  ASM  # 5 org
: frame  ent orgn  S>D ,   >BIN ]ASM ok
```

To erect the bridge, we load a sync boot master into node 300, a bridge loader in 400, and a copy of the bridge code in 500. The code in 400 when activated uses 500 to reset the other chip and 300 to boot 10300 in the other chip, loading it with the bridge code from 500 and activating it. 400 then copies the bridge code from 500 into 300 and activates it. When that is complete, the bridge is fully up and functional, so we appear to have a 288 node chip with two invisible nodes (300 and 10300.)

```
4690
This code is used in setting up the transparent port bridge,
   which will then operate until both chips are reset.
Setup:  Bridge code is in node 500 with a=5, b=io, p=down.
   This code is in node 400 with a=up, b=down, p=0.
   Sync Master code is in node 300 with p=x19 (drives pins low)
Initially, then, 300.17 (clk) and 300.1 (data) are driven low.
@s reads the next word from n500, starting at 5.
!his  sends a word to the other chip's node 300
!ours  stores a word into node 300 RAM at its a, incrementing.
set  stores into node 500's io register.
-rst  and  +rst  drive other chip reset low and high via 500.17
frame  resets other chip for about 5us
```

```
2290
1907 400 ( Bridge Loader)   OVER . ASM[  NODE BIN ERS
# 0 org
: start  ahead
: @s ( - n)  @p !b @b ;  A[ @+ !p ]] ,
: !his ( n)  @p ! ! ;    A[ @p  # 1901 its 18o ]] ,
: !ours ( n)  @p ! ! ;   A[ @p !+ ]] ,
: set ( n)   @p !b !b ;  A[ @p !b ]] ,
: -rst  x20000. set ;   : +rst  x30000. set ;
: frame  then -rst  1000. for next +rst  @s @s @s dup -1. . +
   >r >r >r  !his r> !his r> !his  begin @s !his next
: local  @p !b A[ dup xor a! ]] ,  @p ! A[ # 1901 its off ]] ,
   @p ! A[ dup xor a! ]] ,  x3F. for @s !ours next
   @p ! A[ # 1904 its ent ; ]] ,  @p !b A[ warm ; ]] ,  warm ;

: last   >BIN ]ASM
```

```
4689
This code, in node 300, enables that node to master another chip
   via its sync boot node (300).  It is used during port bridge
   setup to load bridge code into the other chip's node 300, but
   can also be used by IDE to work the other chip as well.
The clock signal (pin 17) is low on reset.  The boot node of the
   other chip sees the pin going high and data on the pin will
   be taken seriously as a boot stream so long as it does not
   stay high too long the first time.  The clock line is high at
   rest.  Receiver samples MSB on first falling clock edge, and
   then each of the next 17 clock edges either direction samples
   another bit.  Clock is high at end of word and stays there
   until the next word begins.
off  drives clock and data low then leaves them WPD which is the
   rest state the bridge code at the other end is looking for.
```

```
2289
1901 300 ( Sync Master)   OVER . ASM[  NODE BIN ERS

# 0 org

: dly ( b)  !b  40. for unext ;  ( ~88 ns)
: 1bt ( b)  dup dly  x10000. xor dly ;
: 18o ( w-0)  x30000. dly  8. for begin
     -while  x30003. 1bt  : rise
       2* -if  x20003. 1bt  2* *next drop ;
         then  x20002. 1bt  2* *next drop ;
     then  x30002. 1bt  rise ;
: off  io b!  x20002. dly  x10001. !b ;

: last   >BIN ]ASM
```

Other erection mechanisms are possible, such as the one Stefan Mauerhofer uses to install this bridge in his Kraken based development environment.