A High Speed Serial Link Interface for SANDAC-V

V. G. Grafe, G. S. Davidson, M. G. Wilde

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

When printing a copy of any digitized SAND Report, you are required to update the markings to current standards.
A High Speed Serial Link Interface for SANDAC-V

V. G. Grafe and G. S. Davidson
Computer Sciences Department 1410
and M.G. Wilde
Command and Control Division 2335
Sandia Laboratories
Albuquerque, New Mexico 87185

Abstract

A need has been identified for high speed communications between embedded computers and their associated gateway computers. To this end, a high speed serial link has been developed in Sandia National Laboratories' Electronics Subsystems Department for interfacing the SANDAC-V embedded computer to an IBM-PC gateway. The resulting 20Mbit/second serial link is shown to be suitable for direct interfaces to sensors, actuators, and other computers in the embedded environment. Characteristics of this link and previous methods are presented. Complete schematics and directions for use are included as appendices.
Contents

1 Introduction 3

2 Prior Approaches 3

3 A Solution 4

4 Applications 7

A SANDAC-V Hardware 10
   A.1 Schematic .......................... 10
   A.2 PLD Equations .......................... 12

B Personal Computer Hardware 14
   B.1 Schematic .......................... 14
   B.2 PLD Equations .......................... 16

C SANDAC/PC File Transfer 17
   C.1 SANDAC Code .......................... 17
   C.2 PC Code .......................... 18
1 Introduction

The SANDAC-V (SANDia Airborne Computer) parallel processor is used in many embedded applications ranging from land based mobile robots to high performance maneuvering re-entry vehicles. Disk and terminal interfaces are not included in the final embedded systems, but are essential during the software development period. This requirement is met by a gateway computer, typically a personal computer. The interface between the gateway machine and the SANDAC has been a low speed (typically 19.2 KBAUD) serial link. The bandwidth of this link is unsatisfactory when rapid transfers of very large files are required, e.g. large images or map databases. This report will describe an interface suitable for these applications developed by the authors while working in the Electronics Subsystems Department. Because this high performance interface makes use of newly available VLSI components, it is also a practical connection for use in the embedded application between the SANDAC, other processors, and sensors and actuators. The strengths and weaknesses of previous approaches are presented in the next section. The characteristics of this link are then presented. Complete schematics and directions for use of the interface are included in the appendices.

2 Prior Approaches

For nearly fifteen years, Sandia has been building and fielding various versions of the SANDAC embedded computer [1]. Relatively low speed terminal communication has been satisfactory until recently. Advances in processor speed and memory density have made possible new applications which require much faster I/O to exchange the large data bases and programs needed to solve problems other than simple control loops [2]. This interface should be bi-directional, transfer at least 300 Kbytes/second (an order of magnitude faster than the existing methods), and should be able to take advantage of higher performance gateway computers as they become available. It should also be suitable for flight line environments which need optical isolation and should have sufficient drive capability for long cable runs (>25 meters). Although serial lines have an obvious advantage in these environments, earlier projects have been forced to use parallel schemes to achieve the desired bandwidth.

One earlier interface used in the Landnav project [3] was based on the IEEE-488 standard [4]. This IEEE-488 bus interface had sufficient bandwidth to support about 100Kbytes/second but was quite complicated to
program and construct. This solution works for short cable runs (less than 10 meters), but is totally unsatisfactory for flight lines.

A unidirectional parallel interface was constructed to support the Hawk operating system program development [5]. This interface was used extensively because it allowed much faster downloading of the test code. This interface has a maximum unidirectional bandwidth of about 33Kbytes/second, which exceeds the 1.9Kbytes/second (19.2KBAUD, 10 bits/byte transferred) afforded by the standard system serial line. The parallel nature of this interface is not suitable for flight lines. The restriction to unidirectional transfers (from the gateway to the SANDAC) is unacceptable for many applications and attempts to extend it to bidirectional operation have not been successful.

The SANDAC hardware supports several serial interface options [6], all of which would be suitable for a flight line. The asynchronous interfaces mentioned above and currently used for console I/O are far too slow. The on-board synchronous channels can run at 4Mbits/second (400Kbytes/second) when supported with external clock generation circuitry. They might be satisfactory but no gateway interfaces or support software exist. Additionally, these onboard channels are often needed for application specific I/O.

3 A Solution

The interface reported here uses the INMOS serial protocol used by the transputer family [7]. This protocol is supported by VLSI chips transferring data at up to 20Mbits/second (about 2Mbytes/second). Earlier work with transputers and these link chips indicated that they would be a good solution to the SANDAC interface problem. It was clear that a high speed serial interface could be readily built with these chips.

An IBM-PC prototyping card was used for the gateway side of the interface. A link adapter chip [7] and associated control circuitry were added to this card. This board is cabled to a similar circuit connected to the SANDAC parallel port as shown in figures 1 and 2. Detailed schematics and example software for these interfaces can be found in the appendices.

The link adapter chip used was an IMSC012 (see Figure 3). The IMSC012 is a universal high speed interconnect, providing full duplex communication according to the INMOS serial link protocol [7]. The link protocol provides synchronized message transmission using byte streams with receiver acknowledge. Data reception is asynchronous, allowing communication to be
Figure 1: High Speed Bidirectional SANDAC-PC Link

Figure 2: Serial Link Implementation
independent of clock phase. The IMSC012 converts the bidirectional serial link data into parallel streams. It can be used to freely interconnect transputers, INMOS peripheral controllers, I/O subsystems, and microprocessors of different families.

Test programs were written to exchange data between the SANDAC and a typical gateway computer (in this case a COMPAQ). Data transfer rates of 68.3Kbytes/second were measured using this gateway. The gateway computer's bus bandwidth was found to be the limiting factor. As a further test, the gateway was replaced by a 15MHz B004 transputer development board [8]. This allowed data transfers of 265 Kbytes/second. This is approximately the maximum sustained data transfer rate of the parallel port with a SANDAC program which checks the link status and then transfers data. This transfer rate is much less than the maximum supported by the link adapter chip. A link adapter serviced by one of SANDAC's DMA (direct memory access) controllers would run about six times faster than the one connected to the parallel port and serviced by a processor.

The simplicity of the concept and ease of implementation is apparent from the figures and appendices. Because of this simplicity, it would be reasonable to use this interface to connect SANDAC to a wide variety of devices. Such devices would be connected with this interface by using just
two twisted pairs of wires, making cabling and optoisolation straightforward. This would mitigate many system integration problems associated with interfacing and cabling.

4 Applications

Current applications for this interface include faster downloading for programs and the operating system. It is also useful for exchanging image data between SANDAC and other computers (the SANDAC gateway, other SANDACs, image processors, etc.). The terrain aided navigation experiments with their associated map databases [9], is another application requiring the exchange of large amounts of data that will benefit from this work.

A camera interface for machine vision is part of a robotics mobility experiment. The interface to this camera is controlled by this kind of serial link. The same interface can now be controlled by a SANDAC computer for any new projects that need a simple camera interface.

Future applications could include on-board disks, high resolution displays, and interfaces to more powerful gateway computers. These simple serial interconnections also will allow powerful distributed embedded computing systems without complicated interfaces and heavy wiring harnesses. Figure 4 shows some connections made possible with these serial links. These interconnections can be reconfigured dynamically using a currently available crossbar switch [10] for even more system flexibility.
References


A SANDAC-V Hardware

A.1 Schematic
A.2 PLD Equations

V. GERALD GRAFE
SANDIA NATL LABS
4/8/87
sandacpal
REV 1
EP600
state machine control for sandac parallel port link interface

PART:
EP600
INPUTS:
clk @ 1.reset1 @ 10,
bdsel @ 3.xasl @ 4,udsl @ 11,ldsl @ 2,rnotw @ 7
OUTPUTS:
berrl @ 19,dtackl @ 18, csl @ 17,q2 ,q1, reset @ 15,
clk10 @ 21, clk5 @ 20

NETWORK:
reset1 = INP(reset1) % par. port reset  
reset = CONF(ireset,) % reset to link IC  
clk10,clk10 = RORF(clk10n,clk,,)
clk5,clk5 = RORF(clk5n,clk10b,,)
clk10b = CLKB(clk10)
clk = INP(clk)
bdsel = INP(bdsel) % address decoder  
xasl = INP(xasl) % address strobe  
udsl = INP(udsl) % upper data strobe  
ldsl = INP(ldsl) % lower data strobe  
rnotw = INP(rnotw) % port read/write  
berrl = CONF(GND,iberr) % error flag  
dtackl = CONF(GND,idtack) % port acknowledge  
csl = CONF(ics1,) % link IC select  
q2,q2 = RORF(d2,clk,xasl,,) % state variable  
q1,q1 = RORF(d1,clk,xasl,,) % state variable

EQUATIONS:
ireset = (resetl');

clk10n = clk10';
clk5n = clk5';

icsl = (q1'); % link IC select
idtack = (q2); % port acknowledge
iberr = (bdsel' * udsl');

d1 = (bdsel' * ldsl' * q2' * q1') % select
    + (q2 * q1); % or read state 1%

d2 = (bdsel' * ldsl' * rnotw * q2' * q1') % read
    +(q1)
    +(q2);
END$
B Personal Computer Hardware

B.1 Schematic
B.2 PLD Equations

V. GERALD GRAFE
SANDIA NATL LABS
4/14/87
PCBDPAL
REV 1
EP600
Miscellaneous logic functions for PC-SANDAC interface

PART:
EP600
INPUTS:
clk20 @ 1, pcreset @ 2, cmdrstl @ 3
OUTPUTS:
clkb2 @ 19, clkb4 @ 18, clkb8 @ 17, clkb16 @ 16,
linkrst @ 4

NETWORK:
pcreset = INP(pcreset) % IBM PC reset
cmdrstl = INP(cmdrstl) % command reset
linkrst = CONF(linkrst,) % reset to link IC
clk20 = INP(clk20)
clk4 = CLKB(clkb2)
clk8 = CLKB(clkb4)
clk16 = CLKB(clkb8)
clkb2,clkb2 = RORF(clkb2n,clk20,,)
clkb4,clkb4 = RORF(clkb4n,clk4,,)
clkb8,clkb8 = RORF(clkb8n,clk8,,)
clkb16,clkb16 = RORF(clkb16n,clk16,,)

EQUATIONS:
ilinkrst = (pcreset + cmdrstl');

clkb2n = clkb2';
clkb4n = clkb4';
clkb8n = clkb8';
clkb16n = clkb16';
END$
C SANDAC/PC File Transfer

C.1 SANDAC Code

* * 
* Program to ship data out the parallel port on SANDAC-V to 
* be stored on a disk on the P. C. host. 
* *
* data equ $80FFC003  * port address to write data 
stat equ $80FFC007  * port address to read status 
* *
* This routine assumes the START address of the block to be moved 
* is in d1 and the END address of the block is in do. 
* *
begin: move.l d1,a0  * get start address into a0 
  lea.l data,a1   * get data port address into a1 
  lea.l stat,a2   * get status address into a2 
  sub.l d1,d0    * calculate the number of bytes 
bra.s loopend   * enter dbra loop at the bottom 
looptop move.b (a2),d2   * get status 
  beq.s loopoptop * while == 0 just wait, still 
busy   move.b (a0)+,(a1)  * copy a byte to output port 
looptend dbra d0,looptop  * branch back for next byte 
  move.b #3,d0  * unless count in do has expired 
  move.b #0,d1  * exit flag 
  trap #4       * "All OK" status flag 
  end          * call GATAR monitor to quit
C.2 PC Code

/* This software handles a transputer port on the P.C. */
/* in input mode only. */
#define DATA 0x300  /* input data port address */
#define STAT 0x302  /* status port for input chan*/
#define Present 1  /* data has been received */
#include "stdio.h"

main(argc, argv)
int argc;
char *argv[];
{
    short Input;
    FILE *fp, *fopen();
    int timeout;
    char Fname[20];
    int done;
    if (argc != 2 )
    {
        printf("
Usage: save filename. ext<cr>\n\n");
        exit(0);
    }
    strcpy( Fname, *++argv);
    if ((fp = fopen( Fname, "r")) != NULL)
    {
        printf("\nFile already exists -- Aborting\n\n");
        close(fp);
        exit(0);
    }
    /* close for read mode and reopen for writing */
    close(fp);
    fp = fopen(Fname, "w");
    done = 0;  /* reset the loop control flag */
    timeout = 0;
    while( ! done )
    {
        Input = inportb(STAT);  /* get links status */
        if( Input & Present )  /* another byte is available */


```c
{
    timeout = 0;
    Input = inportb(DATA); /* get the data byte */
    putc(Input, fp);       /* copy it to the disk file */
}
else
{
    if( ++timeout == 0xFFFF ) done = 1;
}
}
fclose( fp);
printf("\n\nFile %s written and closed\n", Fname);
```
Distribution:
1400  E. H. Barsis
1410  P. J. Eicker
1411  J. L. Akins
1411  T. J. Essenmacher
1411  R. W. Harrigan
1411  M. M. Moya
1412  G. S. Davidson (20)
1412  L. G. Martinez
1412  S. J. Weissman
1413  V. G. Grafe (20)
1413  V. P. Holmes
1413  G. G. Weigand
2110  R. E. Bair
2300  J. L. Wirth
2330  J. H. Stichman
2334  G. M. Heck
2334  S. M. Kohler
2335  K. K. Ma
2335  P. E. Pierce
2335  M. G. Wilde (5)
2335  H. M. Witek
2336  C. R. Borgman
2336  L. J. Dalton
2336  D. L. Harris
2336  J. F. Kobs
2336  N. R. Kolb
2336  B. T. Meyer
2336  E. J. Nava
2336  K. M. Piorkowski
2336  S. C. Richards
2336  H. C. Shen
2336  L. D. Stoll
2340  M. W. Callahan
3141  S. A. Landenberger (5)
3151  W. L. Garner (3)
3154-1  C. H. Dalin for DOE/OSTI (28)
8524  P. W. Dean
9130  R. D. Andreas
9131  D. D. Boozer
9132  A. C. Watts
9133  L. D. Hostetler