Using Multiple FPGA Architectures for Real-time Processing of Low-level Machine Vision Functions

Thomas H. Drayer, William E. King IV,
Joeseph G. Tront, Richard W. Conners

Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University,
340 Whittemore Hall, Blacksburg, VA 24061- USA;
fax (703) 231-3362; phone (703) 231-8657;
e-mail: tdrayer@birch.ee.vt.edu, king@birch.ee.vt.edu,
jgtront@vtvml.cc.vt.edu, conners@birch.ee.vt.edu

Philip A. Araman

United States Forest Service
Brooks Forest Products Center, Virginia Polytechnic Institute and State University,
Blacksburg, VA 24060-0503 - USA,
phone (703) 231-5341

Abstract - **In this paper, we investigate the use of multiple Field Programmable Gate Array (FPGA) architectures for real-time machine vision processing. The use of FPGAs for low-level processing represents an excellent tradeoff between software and special purpose hardware implementations. A library of modules that implement common low-level machine vision operations is presented. These modules are designed with gate-level hardware components that are compiled into the functionality of the FPGA chips. A common input/output interface is created for use in each of the modules, allowing the interconnection of several image processing modules in a parallel or pipelined manner. This new synchronous. unidirectional interface establishes a protocol for the transfer of image and result data between modules. This reduces the design complexity and allows several different low-level operations to be applied to the same input image. A method is developed to partition and compile the design into the hardware resources of multiple FPGA chips. Experimental results verify the efficiency of using common multiple FPGA architectures to implement real-time machine vision processing.**

I. INTRODUCTION

Real-time machine vision systems require not only sophisticated heuristic algorithms custom-suited for a particular application, but also a means of performing mathematically trivial tasks on a massive quantity of input image data. A real-time vision system may require several different low-level processes to be applied to an input image, either simultaneously or sequentially. The low-level processing tasks can require as much or more computational power than high-level vision functions. Hence, special-purpose hardware which is specifically designed to perform early processing functions offers a substantial performance increase. However, traditional approaches to the development of special-purpose hardware are expensive to create and of limited flexibility.

FPGA chips provide large arrays of programmable logic resources that can be reprogrammed an unlimited number of times. Within each chip is an array of Configurable Logic Blocks (CLBs) used to implement sequential or combinational logic. The Xilinx 4000 family of FPGA chips is typical of the high-performance FPGA chips currently available [1]. The CLBs of the Xilinx 4000 family chip contain two flip-flops and two function generators. The function generators create arbitrary logic functions of up to four variables. The functionality of the CLB is established by programming Static Random Access Memory (SRAM) bits within the CLB that control multiplexer for signal routing or look-up tables to define the function generators. Individual FPGA chips have an array of CLBs that are interconnected with programmable interconnection resources. More complicated sequential or combinational logic functions are created by programming the interconnections and logic of the CLBs in the array. Input/Output Blocks (IOBs) allow connection to the pins of the chip. Current FPGA chips have as many as 1024 CLBs, 256 IOBs, and 25,000 equivalent gates [1]. The configuration of Xilinx chips is stored in SRAM residing on the FPGA chip, allowing each chip to be reconfigured many times after initialization. These SRAM-based FPGAs provide a large amount of reconfigurable logic resources.

Using multiple FPGAs to implement special-purpose image processing hardware allows the processing functions to be defined at the gate level, providing a very fast and flexible hardware solution to the problem. Recently, many types of multiple FPGA processing architectures have been developed. Systems such as SPLASH II [2] and the Virtual Computer [3] combine FPGAs and SRAM with switching elements. Other system such as MORRPH [4] and BORG [5] combine FPGAs with fixed interconnections to empty sockets or a prototyping grid. All of these architectures are based on the Xilinx 4000 series FPGA chips. An inexpensive multiple

FPGA architecture provides an excellent resource for the development of real-time machine vision hardware.

Some research has been done on implementing one complex image processing function at a time using a multiple FPGA architecture [6, 7]. However, a typical machine vision task requires several simple low-level operations to be performed on an input image.

However, it is not advantageous to develop each new image processing design from the gate level. This requires the expertise of an experienced digital designer and has the limitation of complicated and time-consuming development. Instead, a library of general-purpose image processing modules is developed. These modules are combined in an arbitrary manner to create complicated image processing designs. This also allows the creation of support tools which decrease the development time for individual modules.

To create a system of flexible modules that can be connected in an arbitrary manner, a common input/output interface to the modules is defined. This provides a method of transferring image and result data between modules. The new protocol is called the Synchronous Unidirectional Image Transfer (SUIT) bus. This interconnect bus is flexible enough to allow a variety of image or result data formats. The bus protocol has been created and refined using knowledge obtained by studying other image bus standards [8, 9, 10] and the creation of flexible image processing modules.

Hence, this paper defines a method of creating image processing designs by interconnecting several low-level image processing modules from an existing library. Several modules are created using the Powerview schematic capture program from Viewlogic Incorporated. Their functionality is verified using simulation capabilities of the Powerview software. Each module is compiled into the resources of a single Xilinx 4010-4 FPGA chip using the Xilinx Design Manager (XDM) software from Xilinx Incorporated. This software provides timing information and FPGA resource utilization used to determine the performance of the modules.

For more complete verification, two complicated image processing designs containing several low-level modules from the library are compiled into a new FPGA-based Custom Computing Machine (CCM), called the MOdular Reprogrammable Real-time Processing Hardware or MORRPH [4]. This CCM contains a 3x2 array of Xilinx 4010-4 FPGA chips in a mesh architecture. Image data is generated by a 864-pixel color linescan camera, the TL-2600 RGB color linescan camera from Pulnix Incorporated [8]. This provides a real-time verification of the image processing hardware development system proposed in this paper.

II. IMAGE PROCESSING HARDWARE DESIGN

Hierarchical techniques are typically used by design engineers to create complex image processing designs. This creates several levels of detail for each design. Experienced digital hardware designers create gate-level schematics at the lowest level. Commercially available schematic capture tools or hardware descriptive languages are used for design entry by these skilled designers.

However, it is desirable to create a development system that allows image processing designs to be created without the specialized knowledge required for gate-level schematic capture design or knowledge of a specialized language such as VHDL. A library of modules with a common interface allows complicated image processing designs to be created without gate-level design. Many commercial image processing hardware architectures provide a similar library of low-level image processing operators. This allows inexperienced designers to create complex designs using the complicated modules created by others.

A. SUIT BUS Format

The SUIT bus uses a global clock to synchronize the transfer of information between image processing modules. This eliminates the timing and circuitry overhead required to generate the handshaking signals associated with asynchronous data transfers. All signals of the bus must meet setup and hold times for this clock signal. Required setup and hold times are defined by the FPGA chips used in the target architecture. For example, when an image processing design is translated into the logic resources of any -4 speed grade FPGA in the Xilinx 4000 family of FPGA chips, the minimum setup and maximum hold times are 4.5 ns and 0 ns, respectively [1]. The XDM software from Xilinx Incorporated allows these timing requirements to be verified after a design is translated into the resources of an FPGA.

Sixteen signals are defined for the SUIT bus, in addition to the global clock. Eight of the signals are data lines **(DATA[7:0])** and the other eight signals are control lines **(DV, CMD[2:0], CSEL[3:0]).** The signal locations and definitions are illustrated in Figure 1.

The channel select **(CSEL[3:0])** lines define sixteen independent channels for the time-multiplexed transmission of image data. Different channels may correspond to separate images, different spectral bands of the same image, or result data.

The data valid signal **(DV)** determines the function of the three command lines **(CMD[2:0]).** When **DV** is low, data is not transmitted on the eight data lines and their values are not defined. The values of the three command lines define the following eight commands. The *marking* command is a used when the bus is idle, denoting a clock period when no information is transferred. The *bus reset* command is used to reinitialize the modules, possibly in the case of an error or system reset. The value of the channel

select signals is not defined for the *marking* command, but is required for all other commands.

Data on the SUIT bus may represent one, two, or three dimensional data. The *line start, frame start,* and *sequence start* commands indicate the start of a l-dimensional line of values, 2-dimensional frame of lines, or a 3-dimensional sequence of frames, respectively. After the entire line, frame, or sequence has been transmitted, *line end, frame end,* and *sequence end* commands are transmitted. These start and end commands are used to frame the data as it is transmitted on the SUIT bus. The channel select signals must be valid during the start and end commands to allow the independent channels to represent data of different dimensions and sizes. For a data structure of any dimension to be transmitted, it must be framed by all three types of start and end commands.

Data values are transmitted on the bus when the **DV** signal is high. The **DV** signal is permitted to be high for ***only one*** clock cycle to transfer a single byte of data. During this clock cycle, the three command lines are used to define properties associated with the current data. The high command bit **(CMD[2])** defines whether the value is contained within an arbitrary Region-Of-Interest (ROI). A separate channel can be used if more than two regions are required for a particular image processing function. The lower two command bits **(CMD[l:0])** define the word size of the data that is transmitted. Data word sizes of one, two, four, or eight bytes are supported by the SUIT bus. Additional bytes of larger data sizes are sequentially transmitted, in order of least significant to the most significant bytes.

Since all data must be completely framed by all six start and stop commands, the sequence of values required to transmit an image on the SUIT bus is determined. However, commands and data for other channels and *marking* commands may be randomly inserted into this sequence. The current definition of the SUIT bus provides a flexible method for interfacing image processing modules,using a minimum number of signals.

B. Image Processing Modules

The definition of a standard interface allows the independent creation of many low-level image processing modules that are easily interconnected. These low-level image processing modules are combined to create complex image processing designs. Design entry for the modules can be accomplished using a number of commercially available schematic capture programs or hardware descriptive languages. Currently, modules are created using the Powerview schematic capture software from Viewlogic Incorporated.

Location	Function
Q15	DV
Q14	CMD 2
Q13	CMD 1
Q12	CMD 0
Q11	CSEL 3
Q10	CSEL 2
Q9	CSEL 1
Q8	CSEL 0
Q7	DATA 7
Q6	DATA 6
Q5	DATA 5
Q4	DATA 4
Q3	DATA 3
Q2	DATA 2
Q1	DATA 1
Q0	DATA 0

SUIT COMMANDS:

If DV = 0

000 - Marking
010 - Line Start
001 - Bus Reset
011 - Line End
100 - Frame Start
101 - Frame End
110 - Sequence Start
111 - Sequence End

If DV = 1

0xx - Outside ROI
1xx - Within ROI
x00 - 1 Byte Data Size
x01 - 2 Byte Data Size
x10 - 4 Byte Data Size
x11 - 8 Byte Data Size

Figure 1. SUIT bus signal locations and commands.

Three performance measures exist for the image processing modules. First, the pixel processing rate of the module must be greater than or equal to the input pixel rate for real-time processing. The 16-bit SUIT bus can only transmit one byte in each cycle of the global clock. To process data from current 512x512 array cameras that produce 30 frames/see, individual modules must operate at clock frequencies above 7.9 MHz. Next, the number of clock cycles required for data to propagate through a module, defined as its latency, is important for some image processing designs, but is not critical to the overall performance of the module. Finally, the amount of FPGA resources required by individual modules is important. The MORRPH board available for module verification contains six Xilinx 4010-4 chips. Each of these chips contains 800 function generators and 800 flip-flops. To reduce the complexity of partitioning the logic of large designs between several FPGA chips, it is required that ail the logic of an individual module must be contained within a single FPGA chip.

The current library of parts is limited, but growing. These modules represent processes required by existing projects at the Center for Automated Processing of Hardwoods (CAPH) at Virginia Tech. The modules are divided into three classes; SUIT bus control modules, FPGA-specific image processing modules, and external logic image processing modules. The SUIT bus control modules provide access to and control functions for the SUIT bus. FPGA specific image processing modules process image data on the bus using only the resources within available FPGA chips. Finally, the external logic image processing modules process image data using the logic of support chips.

1. SUIT BUS Control Modules

The modules defined in this section do not modify the data transmitted on the SUIT bus. Some of the modules provide input or output data formatting to or from the bus. Since the SUIT bus is a synchronous bus that uses a global clock, the SUIT bus is only used to transfer data within FPGAs and between FPGAs on a single printed circuit board. External communication is accomplished using existing image transmission formats such as the Pulnix [8] and Data I/O [10] digital image data transmission formats.

The **PULNIX2SUIT** module translates the Pulnix TL-2600 RGB camera format to the SUIT bus format. This allows input of color image data at rates up to 2.5 Mbytes/sec. Similarly, the **SUIT2PULNIX** module reformats SUIT bus data for output in the Pulnix format. This allows the data to be output for collection by real-time data collection devices. The **SUIT2ISA** module provides a low-bandwidth interface to the Industry Standard Architecture (ISA) computer bus, specifically for the MORRPH board. This module uses ISA bus I/O read cycles to obtain up to 500 KBytes/sec of data from the MORRPH board.

In complicated image processing designs that contain multiple data paths or feedback, additional modules are often required to control the flow of data on each SUIT bus. The 16 separate channels of information available on the bus create the need for these control modules.

The **MULTIPLEX** module combines the data from two separate SUIT busses into a single SUIT bus. Marking cycles are removed from each bus as required to combine the two data streams. The **BLOCK_CHAN** module removes all data and control cycles from one of the sixteen channels and replaces the information with a marking cycle. Information from all other channels is transmitted on the output SUIT bus. The **INC_CHAN** module receives data from all channels on its input bus and transmits each on the next channel of the output bus. Channel sixteen is output on channel zero of the output bus.

The important attributes of these modules are shown in Table 1. This data is obtained by using the Xilinx XDM software to compile the schematic of each module into the resources of a single Xilinx 4010PG191-4 FPGA chip. Table 1 lists the number of FPGA function generators and flip-flops required to implement each module. Signal paths are analyzed to determine the longest propagation path for each module. This determines the maximum frequency of the global clock signal used by each of these modules. The latency of the modules is included for completeness. These attributes illustrate the approximate performance and amount of FPGA resources required to implement these modules.

Table 1. SUIT Bus Control Module Characteristics.

Module Name	Function Generators	Flip-Flops	Speed (MHz)	Latency (Cycles)
SUIT2PULNIX	58	69	10.0	n/a
PULNIX2SUIT	52	51	16.5	n/a
SUIT2ISA	221	155	24.0	n/a
MULTIPLEX	46	101	22.5	2-5
BLOCK_CHAN	16	16	32.7	1
INC_CHAN	3	16	49.9	1

2. FPGA-Specific Image Processing Modules

The modules defined in this section process data of the SUIT bus using only the resources of FPGA chips. These modules perform common low-level image processing functions found throughout image processing literature.

The **THRESHOLD** module uses an 8-bit threshold value to process a single a channel of the SUIT bus. All image values below the threshold are converted to a defined output value. All other image values are not modified. The **COLOR2BW** module creates a black-and-white image from a 3-byte RGB color image on the input SUIT bus by calculating the average of all three color values for each pixel. The **AVERAGE_3xl** module is a 3x1 window operator with unity gain coefficients. This module implements the following function:

$$y(i,j) = [x(i,j) + x(i-l,j) + x(i-2,j)]/3 \quad (l).$$

The **AVERAGE_3xl** module can simultaneously process up to three separate image channels at the same time. A 256x8 RAM location is used to create a look-up table in the LUT module. Each of the 256 possible grayscale values are mapped to new values by the table. The **HISTOGRAM** module calculates a 256 element array of count values. Each count value represents the frequency of a pixel intensity value in an image. Finally, the **LEAD_LAG** module finds the first rising edge and the last falling edge in a line of image data. The characteristics of these modules are summarized in Table 2.

The FPGA-specific image processing modules have a wide variance in their performance and required amount of logic resources. The **THRESHOLD, LUT,** and **LEAD_LAG** modules all require a small amount of resources and operate at a high clock frequency. However, several of the image processing modules require a much slower global clock frequency. Others require almost all of the resources of a large FPGA or were simplified in order to be implemented by a single FPGA chip.

Table 2. FPGA-Specific Image Processing Module Characteristics.

Module Name	Function Generators	Flip-Flops	Speed (MHz)	Latency (Cycles)
THRESHOLD	10	16	26.2	1
COLOR2BW	191	58	6.9	4
AVERAGE_3x1	182	109	6.8	4
LUT	170	16	20.3	1
HISTOGRAM	658	113	31.1	n/a
LEAD_LAG	12	39	64.9	1

The **COLOR2BW** and **AVERAGE_3xl** image processing modules operate at a much slower clock frequency than the control modules. This is a direct consequence of a low-performance 10-bit binary multiply circuit used in these modules.

The 3x1 window size used in the **AVERAGE_3xl** module is significantly smaller than common 3x3, 5x5, or 7x7 window sizes, The amount of memory required for intermediate storage of image data is dependent on the size of the window operator and the size of the image. For typical 512x512 8-bit grayscale images, one Kilobyte of memory is required for a 3x3 window operator. This requires over 700 function generators, consuming 88% of the available resources of a Xilinx 4010 FPGA chip,

Similarly, a large amount of FPGA resources are used to implement memory in the **HISTOGRAM** module. Over 530 of the 658 function generators required for the **HISTOGRAM** module are used to create a large 256x24 (768 byte) memory subsystem.

Even though these modules may not represent the most efficient designs, they do illustrate the problems associated with using FPGAs for real-time image processing functions. The excessive amount of FPGA resources required for operand storage (memory) and the low performance addition and multiply circuits are typical problems associated with using FPGAs for low-level image processing.

3. External Logic Image Processing Modules

The problems of the previous section are solved by coupling memory and arithmetic support chips with the FPGAs. These external support chips implement logic not efficiently realized by the FPGA chips. The FPGA chips provide timing and state machine logic required to implement the modules,

The last four modules of the previous section have been modified by adding external logic to increase their performance. New attributes for these modules after redesign are shown in Table 3. The latency of these modules is unaffected by the redesign, and is not included in Table 3.

The **AVERAGE_3xl** and **COLOR2BW** modules use an external 16x16 Parallel CMOS multiply chip (IDT 7216L from Integrated Device Technology). The 50 nanosecond multiply time of the external multiply chip doubles the operating clock frequency of both modules.

Similarly, external SRAM chips are used to reduce the amount of FPGA logic resources required to implement both the **LUT** and **HISTOGRAM** modules. The use of a single 256Kx32 SRAM chip (MCM 32257 from Motorola) allows the 3-byte RGB color images to be mapped into a 2,000 element color palette by the **LUT** module. The number of function generators required to implement the **HISTOGRAM** chip is reduced by over 80% by using a single 8Kx8 SRAM chip (MCM 6264CP25 from Motorola). At the same time, the size of the histogram is over 8 times larger, creating a histogram for the new 2,000 element color palette. Both of these modules are significantly more complex than the modules presented in the previous section, and cannot be implemented using FPGA resources alone.

A new module is introduced in this section that also uses two external 8Kx8 SRAM chips. The **LITE_COMP** module compensates for irregularities in lighting when using a linescan camera. Each pixel location is linearly transformed using the equation:

$$y(i)j) = m_i x(i,j) + b_i (2).$$

This function also cannot be implemented using a single Xilinx 4010 FPGA chip, and therefore was not included in the previous section. Storing all of the m, and b_i variables in the external SRAM chip allows the design to fit within a single FPGA chip. However, the MORRPH architecture does not provide enough connectivity to allow the use of the external multiplier chip and the two SRAM. The 10-bit multiply circuit used by the **LITE_COMP** module reduces the maximum clock frequency to 6.7 MHz.

The **LITE_COMP** module illustrates the drawback associated with using external memory or arithmetic chips. Instead of being constrained by available logic resources of the FPGA's, the modules become constrained by the I/O resources of the FPGAs. Careful consideration of the

Table 3. External Logic Image Processing Module Characteristics.

Module Name	Function Generators	Flip-Flops	IOBs	Speed (MHz)
COLOR2BW	90	48	64	14.3
AVERAGE_3x1	86	83	64	14.4
HISTOGRAM	127	113	28	15.0
LUT	9	79	64	44.0
LITE_COMP	174	109	56	5.5

performance requirements of each design is required when making choices for image processing designs.

III. RESULTS

The modules defined in the sections above have been combined to create two complicated image processing designs. These designs are shown in Figures 2 and 3. The image processing design of Fig. 2 creates both a grayscale histogram and a histogram of the color image after it has been mapped into a palette of 2,000 colors. The image processing design of Fig. 3 performs three successive average operations on an image, creating a 7 x 1 window operator.

Both designs have been compiled and verified on the MORRPH architecture. The image processing design of Fig. 3 is compiled into the resources of a single Xilinx 4010, while the design of Fig. 2 requires six Xilinx 4010 chips. Both designs have been verified at 16 MHz, faster than the typically conservative estimates of 13-14 MHz generated by the Xilinx XDM software.

IV. CONCLUSIONS

FPGAs have been shown to provide an excellent resource for creating real-time image processing hardware. The gate-level resources of FPGA chips allow arbitrary designs to be created from a library of low-level image processing modules. However, additional memory and/or arithmetic support chips are often required to achieve the desired speed and utilization performance for real-time operation of complicated modules. FPGA-based custom computing machines that allow arithmetic and memory chips to be tightly coupled with each FPGA, such as the MORRPH board, provide the most efficient and flexible architectures for real-time image processing.

V. ACKNOWLEDGMENTS

This work has been funded in part by the Southeast Forest Experiment Station of the United States Forest Service and the Bradley Department of Electrical Engineering of Virginia Polytechnic Institute and State University. Thanks also to Peter Athanas, Earl Kline, and Paul Lacasse.

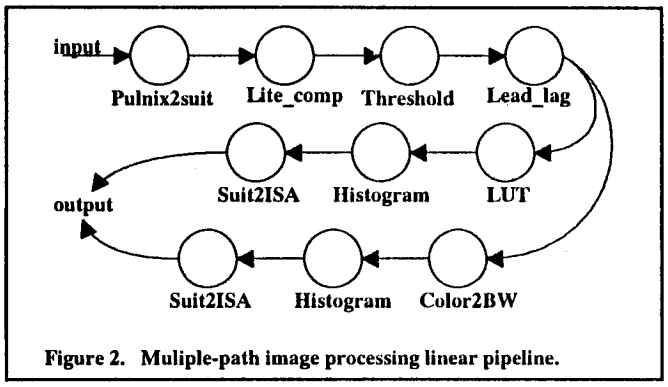


Figure 2. Muliple-path image processing linear pipeline.

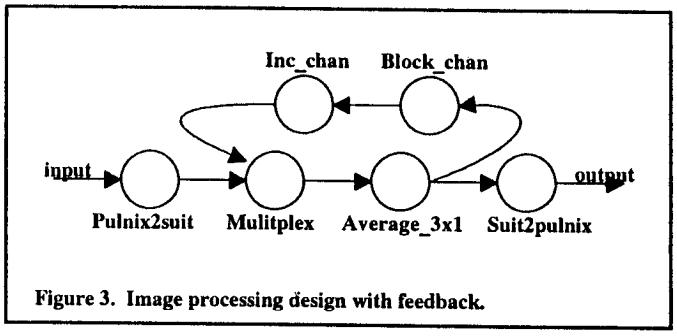


Figure 3. Image processing design with feedback.

VI. REFERENCES

[1] Xilinx Corporation, "The Programmable Logic Data Book," 1994.

[2] Arnold,J.M., Buell, D.A., and Davis, E.G. "Splash 2," in Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures, 1992, pp. 316-322.

[3] Casselman,S., "Virtual Computing and the Virtual Computer, " in Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines (FCCM '93), Napa CA, April 1993, pp. 43-48.

[4] Drayer, T. H., Tront, J. G., King, W.E, et. al., "MORRPH: A MOdular and Reprogrammable Real-time Processing Hardware", in Proc. of FCCM '95, Napa CA, April 1995.

[5] Chan, P. K., "A Field Programmable Prototyping Board: XC4000 BORG User's Guide," UCSRC-CRL-94-18, April 1994.

[6] Abbott, A. L., Athanas, P. M., Chen, L., and Elliot, R. L., "Finding Lines and Building Pyramids with Splash 2", in Proc. of FCCM '94, Napa CA, April 1994, pp. 155-161.

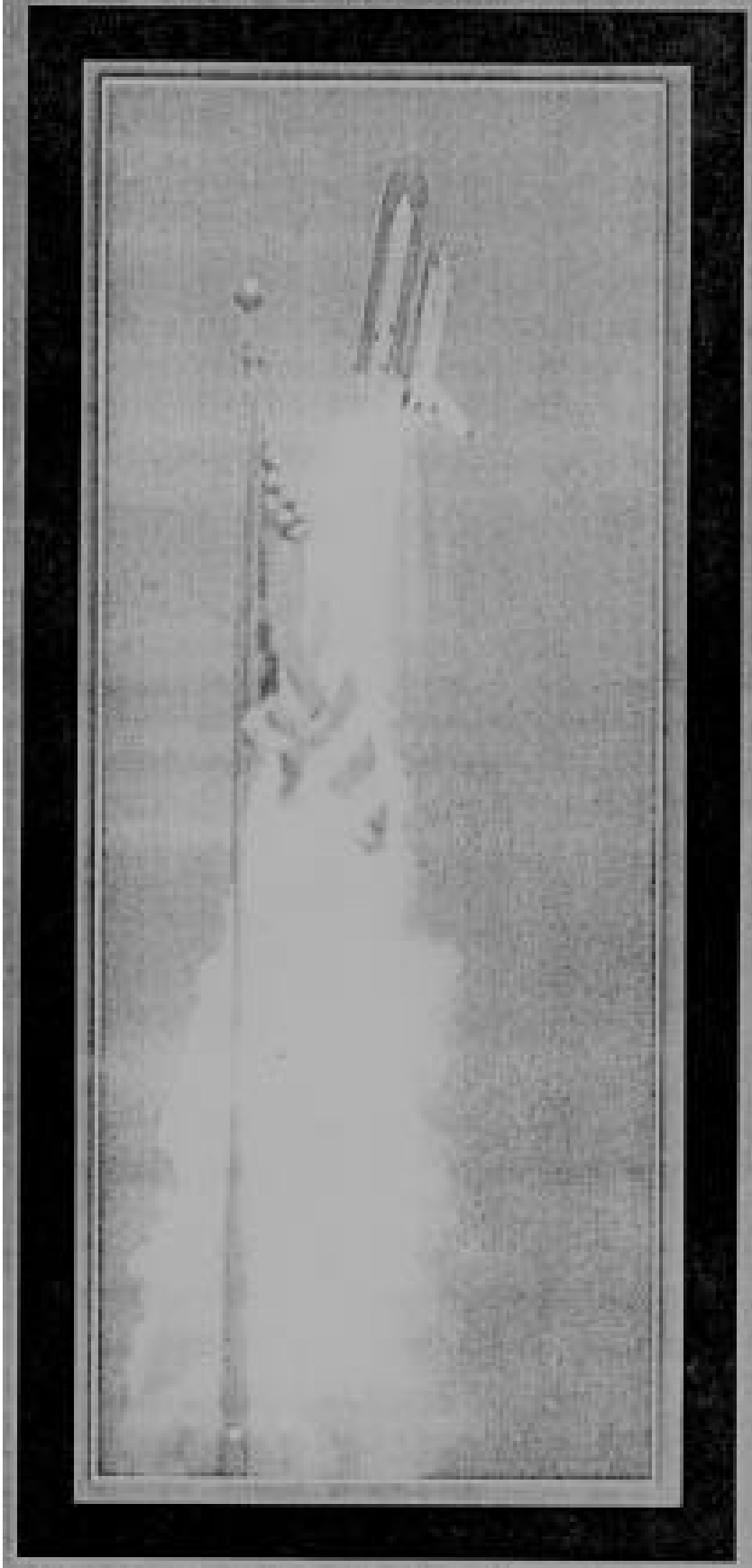
[7] Ratha, N. K., Jain, A.K, Rover,D.T., "Convolution on Splash 2", Proc. of FCCM 95, Napa, CA. April 1995.

[8] Pulnix, Inc., "TL-2600 RGB Linescan Camera Operating Instructions," 1987.

[9] Datacube Inc. "MaxVideo MAXbus Specification," Dec. No. SP00-5, September, 1988.

[10] Data Translation, Inc., "DT-Connect II Bus Specification," 1992.

IECON '95



1995 IEEE 21st

International Conference on Industrial Electronics, Control, and Instrumentation

Volume 2 of 2

Signal Processing & Control
Robotics Vision & Sensors
Emerging Technologies
Factory Automation

November 6 - 10, 1995

Sponsored by

The Industrial Electronics Society (IES) of the IEEE
The Society of Instrument and Control Engineers of Japan (SICE)