24.422 SWITCHING & AUTOMATA THEORY

LABORATORY 4

DESIGN OF A GRAPHICS PROCESSOR: PART 2: THE CCU

PURPOSE & OBJECTIVES

The purpose of this lab is to implement the Computing Control Unit (CCU) of a simple graphics processor called **GRAFIX**. In Lab 5, GRAFIX will be finalized by assembling the DPU and CCU and, possibly, implementing it on a Xilinx FPGA.

The objective is to learn how to

(a) formulate the CCU interfacing with the GRAFIX I/O and with the DPU,
(b) formulate a Command Interpreter,
(d) formulate an Controller/Sequencer, and
(c) implement, verify and simplify the CCU.

MAJOR ACTIVITIES

1. Study the modifications to the general GRAFIX architecture from Lab 3.
2. Study the command format of the GRAFIX processor.
3. Complete the architecture of the CCU unit.
4. Implement and verify the Command Interpreter.
5. Implement, verify and simplify the Controller/Sequencer.
6. Implement and verify the CCU.
7. Follow the instructions of TA in the lab.

TIME TO COMPLETE: 3 hours

ANTICIPATED BACKGROUND

1. 24.222 Logic Circuits.
2. 24.361 Microprocessing Systems.
3. 24.424 Microprocessor Interfacing.
4. 24.423 Digital Systems Organization.

RESULTS

An optimal CCU in GRAFIX.

LAB REPORT

See Section 4 of this lab writeup.

ACKNOWLEDGEMENT

Jonathan Greenberg, **Alexis Denis**, and **Luotao Sun** have helped in the design and preparation of this laboratory.

1. INTRODUCTION

The purpose of this lab is to implement the Computing Control Unit (CCU) of a simple graphics processor, GRAFIX.

2. DESCRIPTION OF GRAFIX

2.1 Description of GRAFIX

As described in Lab 3 and illustrated in Fig 1, the GRAFIX processor can perform two different basic operations: (a) drawing an individual pixel, and (b) drawing a line using the Bresenham line drawing algorithm. These commands are issued to the GRAFIX processor through the 8-bit input bus (I-Bus) in a serial manner. Commands are buffered by the host processor and are sent when GRAFIX informs the host that it is ready. When the WAITING_FOR_COMMAND line is set to high, the host processor will begin transmitting the next available command to GRAFIX, sending one byte per clock cycle. Commands transmitted to GRAFIX before it is ready will be ignored and, therefore, GRAFIX cannot be interrupted until its current executing command has completed.

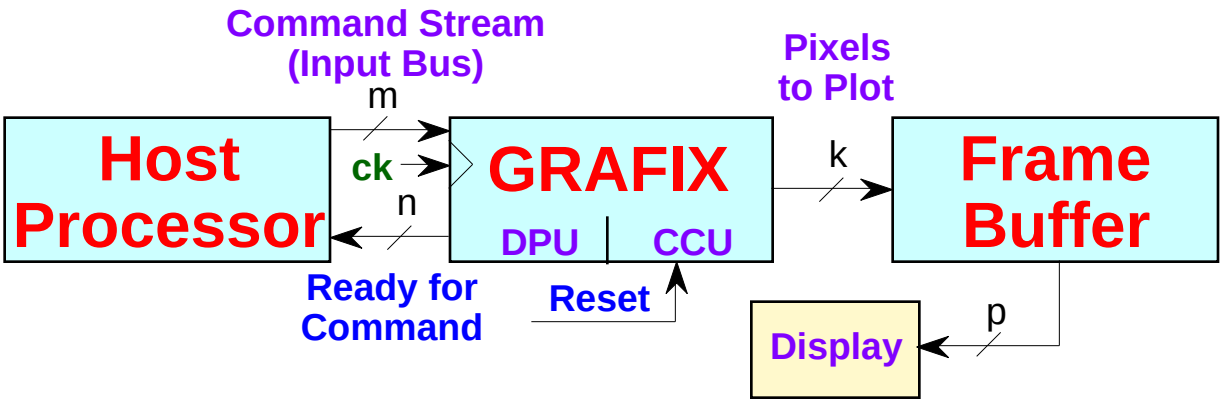

Fig. 1. A host controls a display through GRAFIX and a frame buffer.

Table 1 describes the commands currently supported by GRAFIX.

Table 1: GRAFIX commands.

COMMAND	DATA FORMAT
Sit Idle NUL	(ASCII #00)
Draw Pixel	P Xc Yc Colour
Draw Line	L Xs Ys Xe Ye Colour

As an example, Table 2 lists the sequence of bytes sent on the input bus that will command GRAFIX to plot a single point at (56,79) with a grayscale of 128. Notice again that commands are sent one byte per clock.

Table 2: Input command format.

CLOCK CYCLE	DATA ON INPUT BUS
…-1	NUL
0	'P'
1	0x38
2	0x4F
3	0x80

Output to the frame buffer is controlled using the VALID_OUTPUT control signal. When VALID_OUTPUT is set high, the frame buffer will load from the 24 bit output bus (O-Bus) the x, y coordinates and the colour of the next point to draw. The design is simplified by not requiring the construction of address information from raw coordinate information. Thus pixels can be plotted simply by loading values X, Y and Colour on the output bus, and informing the frame buffer that a valid pixel is ready.

The overall GRAFIX diagram is shown in Fig. 2.

2.2 Description of the DPU

The DPU is the same as described in Lab 3. The ALU was implemented in Lab 3 and the rest of the DPU will be constructed in Lab 5.

2.3 Description of the CCU

If the DPU is the arm of the processor, the CCU is its brain. The CCU controls the sequencing of the events internally and externally according to the information received from the GRAFIX interface.

Internal events are:
- Select the required ALU function.
- Select internal buses for input and output.

External events are:
- Determine the value of VALID_OUTPUT.
- Determine the value of WAITING_FOR_COMMAND.

Information received:
- Command through the I-Bus.

As data bytes are sent down the input bus, they are loaded into the input register Ri. The value of Ri is then passed through the Command Interpreter which determines (via some combinational logic) which of the three commands is being issued. The Command Interpreter output determine the next state of the Control Sequencer, selecting the branch of the state diagram corresponding to the specific command.

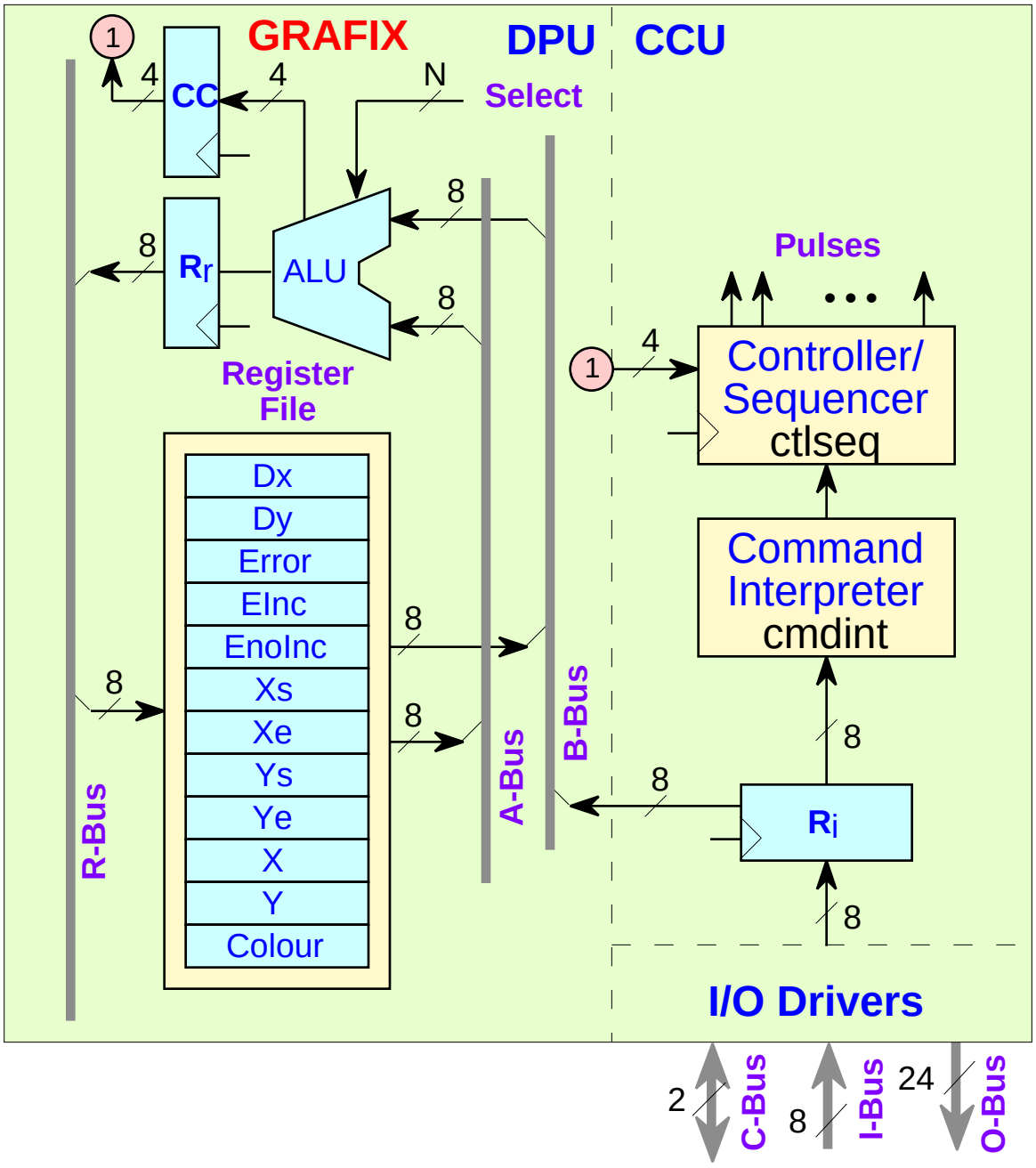

Fig. 2. Block diagram of GRAFIX.

The Instruction Register

The instruction register R_i is necessary because we do not know if the input provider (the host) can hold the lines stable for the entire duration of the clock cycle when both the instruction interpretation and execution sequencing occur.

3. PROBLEMS

3.1 Problem 1: Grafix Architecture

Investigate the block diagram of Fig. 2, and make the following decision regarding the machine specifications:

(a) Decide how many lines are necessary to control the A-Bus and fill in Table 3.
(b) Decide how many lines are necessary to control the B-Bus (note that Ri is connected to the B-Bus).
(c) Decide how many lines are necessary to control the R-Bus input.
(d) Decide how many lines are necessary to control the R-Bus output.
(e) Determine the output size and values of the Command Interpreter.
(f) Determine the inputs of the Controller/Sequencer.
(g) Determine the outputs of the Controller/Sequencer.
(h) Determine the interface between the CCU and the DPU. Specify the type of flip flops used in the CCU and the DPU (e.g. rising edge, data lockout). Illustrate through a clocked example the sequence of events used to pass the X coordinate received from the input bus to the X register.

Table 3: Register selection codes for the A-Bus.

REGISTER SELECTION CODE	REGISTER NAME
	Dx
	Dy
	Error
	Einc
	EnoInc
	Xs
	Xe
	Ys
	Ye
	X
	Y
	Colour

3.2 Problem 2: Command Interpreter in Verilog

Implement and verify the Command Interpreter of GRAFIX using Verilog.

3.3 Problem 3: Controller/Sequencer (Point Function)

Design and draw the branch of the state diagram required to perform the Draw Pixel command (see Fig. 3, the input bits and output bits have to be specified). Note that to reduce clutter, only input/output relevant to a particular state transition need be specified. Don't Care Conditions are not to be listed in state transitions.

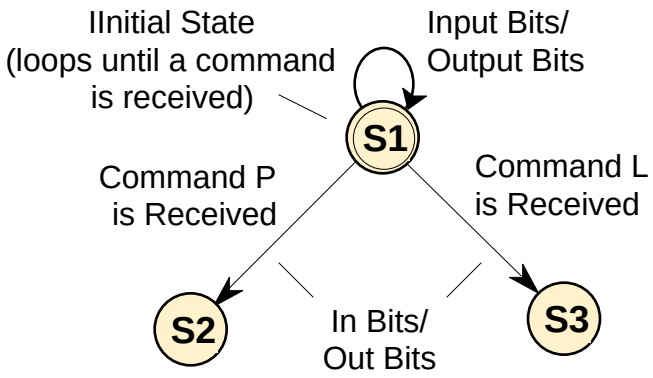

Fig. 3. The beginning of a state diagram for the Controller/Sequencer.

3.4 Problem 4: Controller/Sequencer (Line Function)

Design and draw the branch of the state diagram required to perform the Draw Line command. Consider the following questions:

(a) How many states were required to specify both branches?
(b) How many flip-flops would be required to store these states?
(c) Were all the registers in the table necessary, and if not, how could the table be reduced?
(d) Without changing the external specifications or the speed of the internal clock, what design modifications could be made to the DPU.
(e) Estimate and briefly explain the possible speedup that would result from these optimizations.

3.5 Problem 5: Controller/Sequencer in Verilog

Implement the entire Controller/Sequencer in Verilog.

3.6 Problem 6: CCU in Verilog

Implement the complete CCU in Verilog.

4. REPORT

- Describe the additions and choices for Problem 1.
- Provide a source code for the CCU.
- Provide a state diagram with comments on the event happening at each edge for Problem 3 and 4.

- Can you answer the questions in the POINTS TO PONDER section?
- Have you reached the objectives of this lab?

REFERENCES

[HaSo96] Gary D. Hachtel and Fabio Somenzi. *Logic Synthesis and Verification Algorithms.* Boston, MA: Kluwer, 1996, 564 pp. (ISBN 0-7923-9746-0)

[Kins99] W. Kinsner. *Switching & Automata Theory*. Course Notes for 24.422, Winnipeg MB: University of Manitoba, 1999.

POINTS TO PONDER (LAB 4)

GRAFIX

1. What kind of security mechanism would you include to avoid the CCU from getting stuck in a particular state?

2. Can the Bresenham algorithm be parallelized? If yes, how? Describe the changes that would be made to the GRAFIX unit.