FPGA-Scope: A Labkit Implemented Oscilloscope
6.111 Final Project Proposal

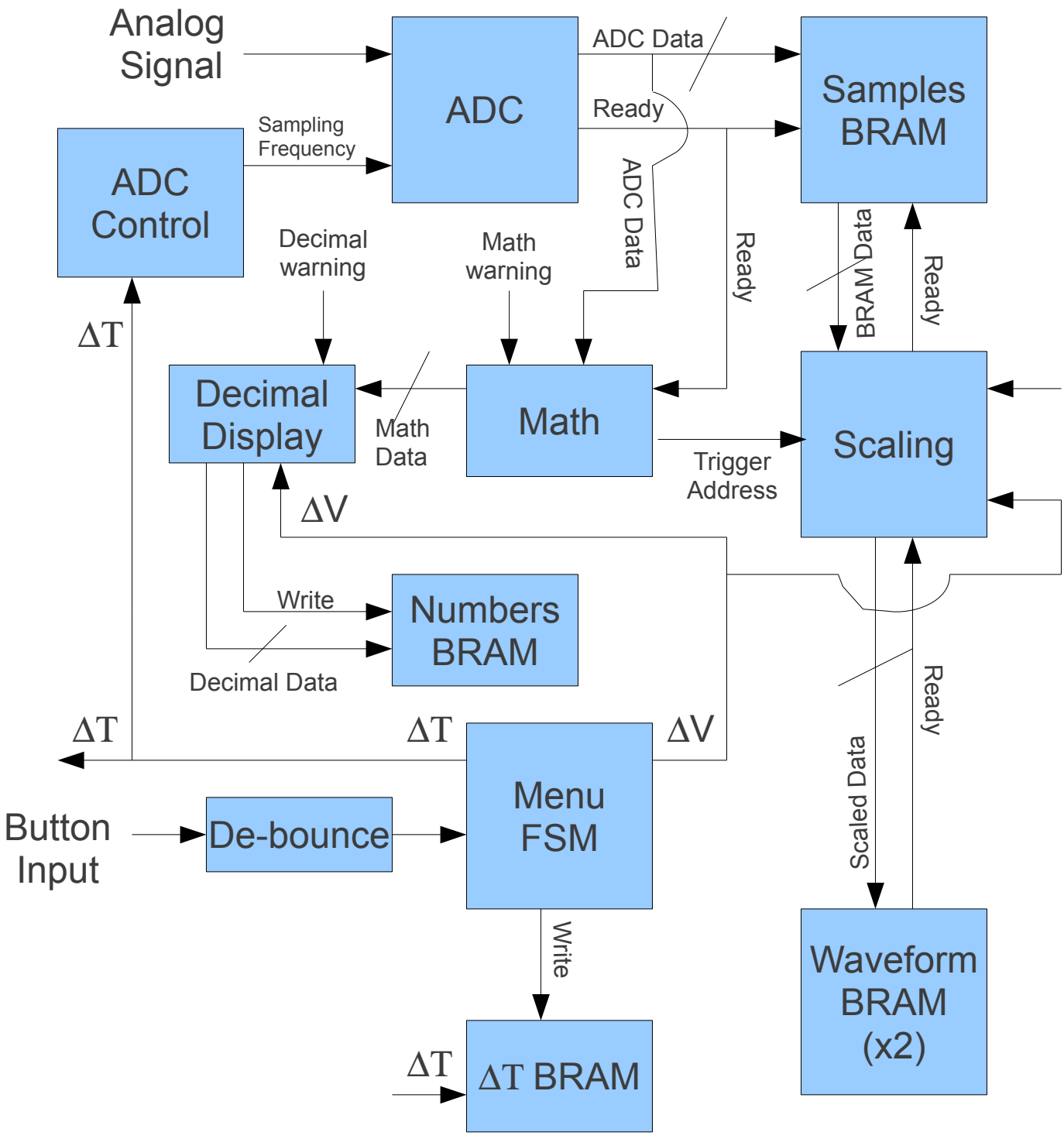
Anartya Mandal and Kevin Linke

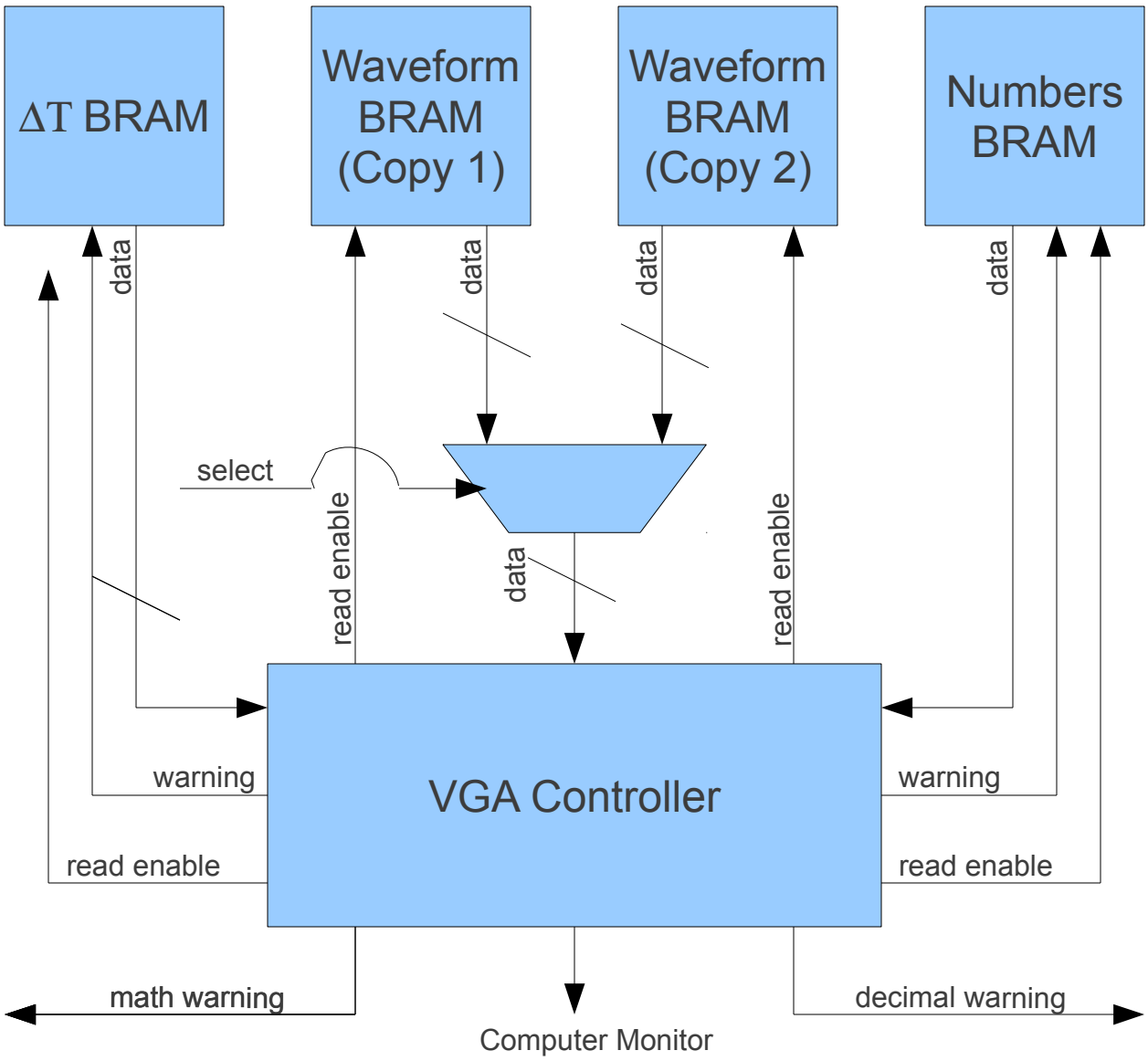
November 1, 2011

1 Overview

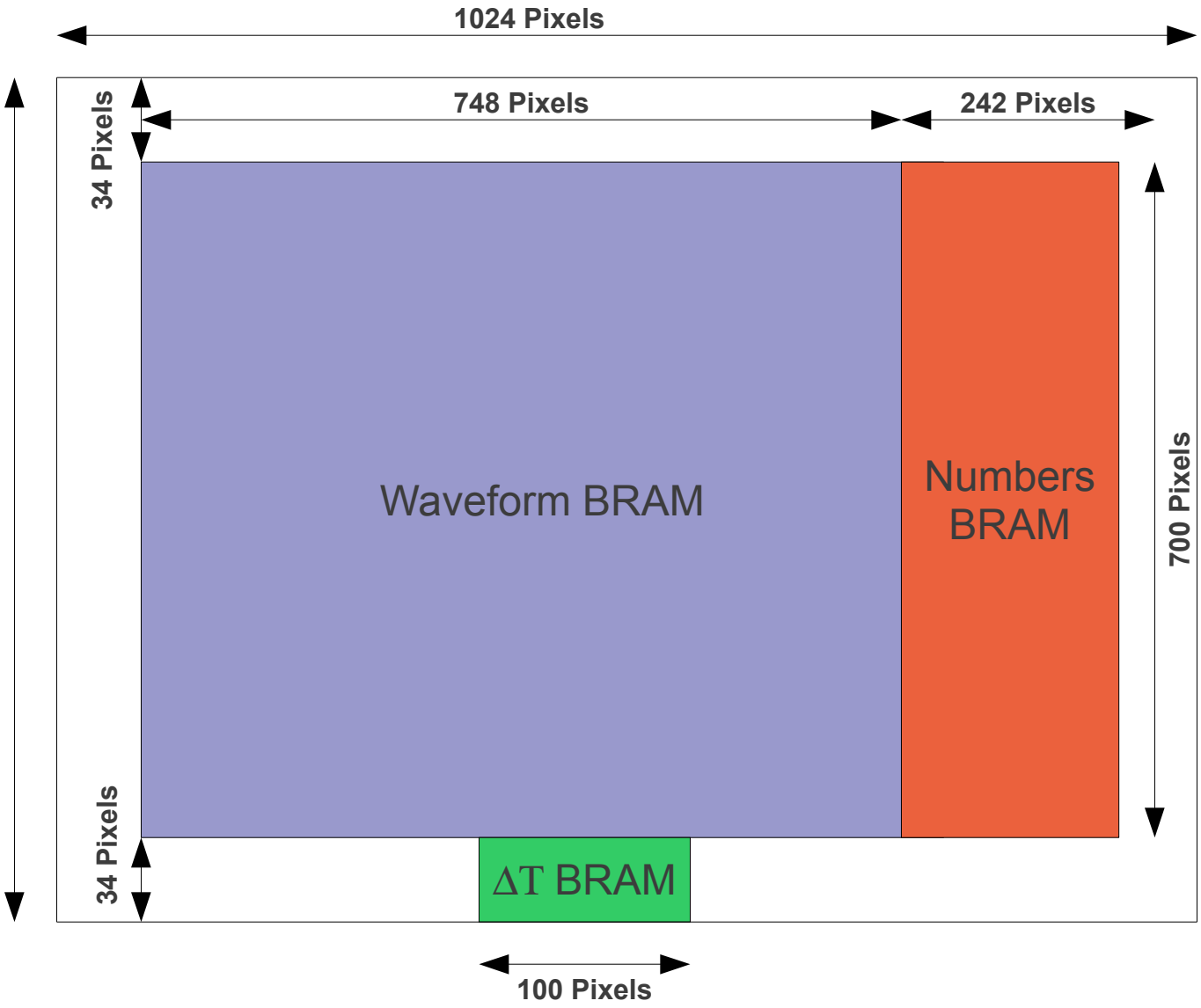
Our final project will be a digital oscilloscope implemented on the Labkit's Field Programmable Gate Array with a computer monitor as the display. In order to accomplish this, we will use a analog-to-digital converter to sample an analog user input. We will store these samples in the FPGA BRAM and then extract information about the input signal from them, such as period, peak-to-peak voltage, and offset. This information will be used to scale the waveform vertically and horizontally such that it can be stored in a separate BRAM array that represents the pixels on the monitor. The scale of the waveform on the monitor will be set via a user interface through the buttons on the Labkit. Also added to this BRAM will be images of numbers that will provide the user with numeric measurements from the waveform, as well as grid-lines to provide scale.

One advantage of the FPGA-Scope is that it would provide a much higher level of scalability than other oscilloscopes. Improving bandwidth and accuracy would simply be a matter of using a better ADC, FPGA or BRAM. This flexibility would avoid the huge costs of purchasing new oscilloscopes. A further feature that might differentiate the FPGA-Scope from other oscilloscopes is the number of signals it can measure. Most oscilloscopes only have four probes, but memory and time permitting, we may be able to add up eight analog inputs to the FPGA-Scope, which would be useful for devices with many outputs, such as an EEG.


Analog Signal
ADC
ADC Data
Samples BRAM
Ready
Sampling Frequency
ADC Control
ADC Data
Ready
BRAM Data
Ready
Decimal warning
Math warning
ΔT
Decimal Display
Math
Math Data
Scaling
Trigger Address
ΔV
Write
Numbers BRAM
Decimal Data
ΔT
ΔT
ΔV
Menu FSM
Button Input
De-bounce
Scaled Data
Ready
Write
Waveform BRAM (x2)
ΔT
ΔT BRAM


ΔT BRAM
Waveform BRAM (Copy 1)
Waveform BRAM (Copy 2)
Numbers BRAM
data
data
data
data
select
read enable
data
read enable
VGA Controller
warning
warning
read enable
read enable
math warning
Computer Monitor
decimal warning

Computer Monitor:



2 Data Collection (Anartya)

Data collection in the FPGA-Scope is performed by a group of three modules: the ADC, ADC controller, and samples BRAM. Together, they sample the analog input at a user-specified frequency, then store the samples in a BRAM for later scaling and display.

2.1 ADC

We will be using an AD574 12-bit analog-to-digital converter to sample the user input. The sampling frequency will be determined by the output of the ADC controller. The status output of the ADC will be fed as a write enable signal to the samples BRAM along with the 12-bit data samples. The speed of the ADC will limit the maximum frequency of the input signal to our oscilloscope; since the max sampling frequency of the AD574 is 10kHz, our max input frequency will be 1kHz to provide ten samples of display resolution per period.

2.2 ADC Controller

The ADC controller takes as an input the delta-t set by the menu FSM, and produces a control signal of the appropriate frequency that tell the ADC when to sample the incoming signal. For lower frequency inputs, samples need to be collected less frequently, because only 748 samples can be displayed on the monitor, but multiple periods of the signal need to be sampled.

2.3 Samples BRAM

The samples BRAM stores the raw data collected by the ADC. This data is stored here until it is rescaled by the scaling module and sent to the waveform BRAM for display. Although the screen will only display 748 samples at a time, the BRAM must store four times this many samples, for a total size of 748*12*4<2^16 bits. The reason for this oversampling is that in order for the waveform to remain in a constant location on the display, we must begin sampling it at a consistent trigger point. Sampling for four times the width of the display window will allow us to search through the samples collected for a trigger condition, then display the samples in the window that surrounds this point. At first we will use a simpler method of triggering, such as peaks, but hopefully we will eventually use a more advanced method such as crossing a threshold.

3 Data Processing

Data processing in the FPGA-Scope takes raw binary data from data collection and user input modules and converts it to a form can be displayed on the screen. Much of this processing occurs in the math and scaling modules, which extract the useful information from the analog input and generate a waveform image for display. Other processing occurs in the decimal module, which converts binary numbers to images of decimals that the user can read. The images created by this processing are stored in the numbers and waveform BRAMs.

3.1 Math Module (Anartya)

The math module is responsible for processing the ADC data on its way into the samples BRAM. This processing includes measuring statistics about the input signal for the user. The three main parameters of interest are average voltage, peak-to-peak voltage, and frequency. Average voltage will simply be calculated using a running sum of weighted values. Peak-to-peak voltage will be calculated by keeping track of the largest and smallest voltage samples encountered. Frequency will be determined by recording the times between two maxima. The math module also has the role of determining the trigger address and sending it to the scaling module. The values calculated by the math module are sent to the decimal module for display.

3.2 Decimal Module (Kevin)

The decimal module receives statistics about the input signal from the math module, as well as the delta-V setting from the menu FSM, and converts them to a decimal display that is readable for the user. The decimal module will use a binary-to-decimal converter to select pre-created images of numbers, then store them in the numbers BRAM along with pre-created image labels.

3.3 Numbers BRAM (Kevin)

The numbers BRAM converts data the math module to a format that can be displayed by the VGA. The image is 700 by 242, so 2^18 bits are required to store it. Like the other BRAMs used by the FPGA-Scope, the numbers BRAM is one-dimensional in order to conserve memory.

3.4 Scaling Module (Anartya)

The scaling module takes the input samples BRAM and converts it to the image that will be stored in the waveform BRAM and then displayed on the screen. Based on the delta-V and delta-t specified by the user, scaling module scales the sample data vertically and horizontally. This feature allows the user to make the waveform fill up the screen appropriately. The scaling module also uses the trigger address from the math module in order to determine which samples from the samples BRAM are actually displayed. Furthermore, the scaling module adds fixed grid-lines to the image in the waveform BRAM in order to provide scale for the user.

3.5 Waveform BRAM (Anartya)

The waveform BRAM stores the image of the waveform created by the scaling module for display by the VGA. Unlike the other BRAMs, we will use two copies of the waveform BRAM. The VGA display requires an image to be read from memory 60 times per second, but the waveform BRAM will take longer than 1/60 seconds to fill. Thus, one BRAM will store the previous copy of the image for reading by the VGA, while the other is being written to by the scaling module. The total memory required will be 700*748*2<2^20 bits.

4 User Interface (Kevin)

The user interface of the FPGA-Scope allows the user to control the horizontal and vertical

scaling of the displayed waveform. It takes user input from the Labkit buttons, and uses it to control the data collection and data processing modules.

4.1 Menu FSM

The menu FSM takes debounced button inputs from the user and specifies the delta-V and delta-t parameters for the ADC controller, decimal module, delta-t BRAM and scaling module. Button one will specify that delta-t is to be changed, and button two will specify that delta-v is to be changed. Once either button is pressed, the up and down buttons will be used to change the parameter values. The menu FSM is also responsible for creating the image of delta-t that is stored in the delta-t BRAM.

4.2 Delta-T BRAM

The delta-t BRAM contains an image of the time scale of the waveform, plus a label. Its size is 34*100<2^12 bits.

5 VGA Display (Kevin)

5.1 VGA Controller

The VGA controller combines the numbers BRAM, the delta-t BRAM and the waveform to create a signal that is displayed on the computer monitor. Each of the BRAMs contains an image that is ready for display on the screen, but they must be positioned relative to each other and combined. The VGA controller also provides read/write timing for other modules. Since the BRAMs must be read 60 times per second, the VGA controller needs to send out a signal (write warning) to warn the decimal module and the menu FSM not to write when it is reading. The VGA controller also produces a select signal that controls which waveform BRAM is being written to/read from.

6 Memory Usage

The total memory usage of the FPGA-scope is 2^16 + 2^18 + 2^20 + 2^12 = 1.38 Mb. This is significantly less than the memory built in to the FPGA, 2.9 Mb. If we have enough excess memory and time, we may be able to implement multiple channels for the oscilloscope.

7 Testing

Since we hope to write the code for the data acquisition/processing and image display in parallel, testing the FPGA-scope will be more difficult than if we developed the modules sequentially. Each module will require an extensive test jig in order to be tested in ModelSim. For instance, testing the scaling module will require a fake samples BRAM filled with pre-set data. Likewise, testing the math module will require a test stream of input data. Testing of the control, timing, and basic logic signals for the modules that write display BRAMs can be done in ModelSim, but

in order to check whether the entire contents of the BRAM is correct, we will have to display them on the monitor. Furthermore, since the ADC is a hardware component, it will have to be tested using a function generator and logic analyzer. Once each of the modules has been tested individually, we will begin to interconnect them. First we will combine the data acquisition/processing modules, then the display modules, then unite the two halves of the project. At this stage, it will be necessary to test the FPGA-scope using a logic analyzer. This device will allow us to test a variety of input patterns (square wave, sawtooth, sine wave) at a variety of frequencies (up to the max frequency of the FPGA).