Roberto Carli
Alessandro Yamhure

6.111 Project Proposal
11/04/2005

MUSIC TRANSCRIBER

Overall System Description

The aim of this digital system is to convert music played into the correct sheet music. We are basically implementing a music transcriber that transcribes sound inputs into visual sheet music on a computer monitor.

The system is divided into three main modules:

The Note Recognizer is responsible for analyzing the sound input of the system and outputting its note name (F, C#, etc) and rhythmical duration. It takes in one input (the audio) and gives out four outputs; *note* (the name of the note), *duration* (how many beats it lasts), *new note* a pulse that signals the arrival of a new note to the **Video Display** module and *beat,* a pulse emitted at the tempo of the music.

The Metronome is responsible for converting a pulse signal *beat* into a rhythmical "click" audio sound, which will be fed to the audio codec.

The Video Display module is responsible for taking the *note* and its *duration* when *new note* goes high and converting it into the appropriate video display of staves and notes on the computer screen.

Note Recognition Module (*Alessandro Yamhure*)

The main input is an audio input that comes into the module from a microphone or amplified instrument and passes through a *codec* module which amplifies the audio and converts it from analog to digital.

PITCH DETECTION:

The digital *audio data* is then temporarily stored in RAM before serving as the input to the **FFT** submodule. The **FFT** submodule takes the Fast Fourier Transform of this data and temporarily places its results into RAM memory, before it is used by the **Peak Detector**. The result of the **FFT** is basically a frequency plot of the audio input.

The **Peak Detector** is responsible for using the frequency plot in RAM to detect the peak frequency of the *Frequency Data Out* and in this way detects the pitch of the input.

The **Look-up Table** submodule uses the *peak frequency* to decide the actual musical name for that frequency (F#, Bb, etc). The *Note* serves as one of the outputs of the Note Recognition Module but also serves as the input to the **compare** submodule.

RHYTHM DETECTION:

The **compare submodule** compares the current *note* to the old *note* that was output during the previous processing cycle by the **look-up table**. If they are equal, *change* signal is low. If the two values are different, then *change* is high. If *change* is high we have a "new note" and the signal *New Note*, which is an output of the Note Recognition module, goes high for one clock cycle. It also serves as an input to the **counter** submodule.

The **counter** submodule is a counter that gets incremented every time it is sent a pulse by the **Tempo Clock.** When *change* is high, counter outputs the value up to which it has counted as the *duration* of the note which is another output of the Note Recognition module.

Tempo Clock appropriately divides the system clock (using a counter) into the beats of the music (like a metronome). This is important in order to measure the *duration* of a note, which will allow us to have a rhythmical component in our transcriber.

Video Display (Roberto Carli)

The video display module will convert the digital signals giving information about notes and durations into an actual music sheet, which will be displayed. This module will take three inputs from the note recognizer: the value of the *note*, its *duration* and the *newnote* signal (display another note).
The way I am planning to tackle the display is by using an object-oriented visualization style. The duration of the note will determine its shape (full, quarter etc…) while the value of the note will determine its position on the stave. The video display will have several submodules:

Tracker: The tracker submodule will be an FSM whose function is keeping track of the position of the latest note and, when a new note has to be displayed, to determine what position it will occupy on the screen. Tracker will take inputs *note* and *newnote*, and will output the screen coordinates *cx* and *cy* where the note shall be displayed.

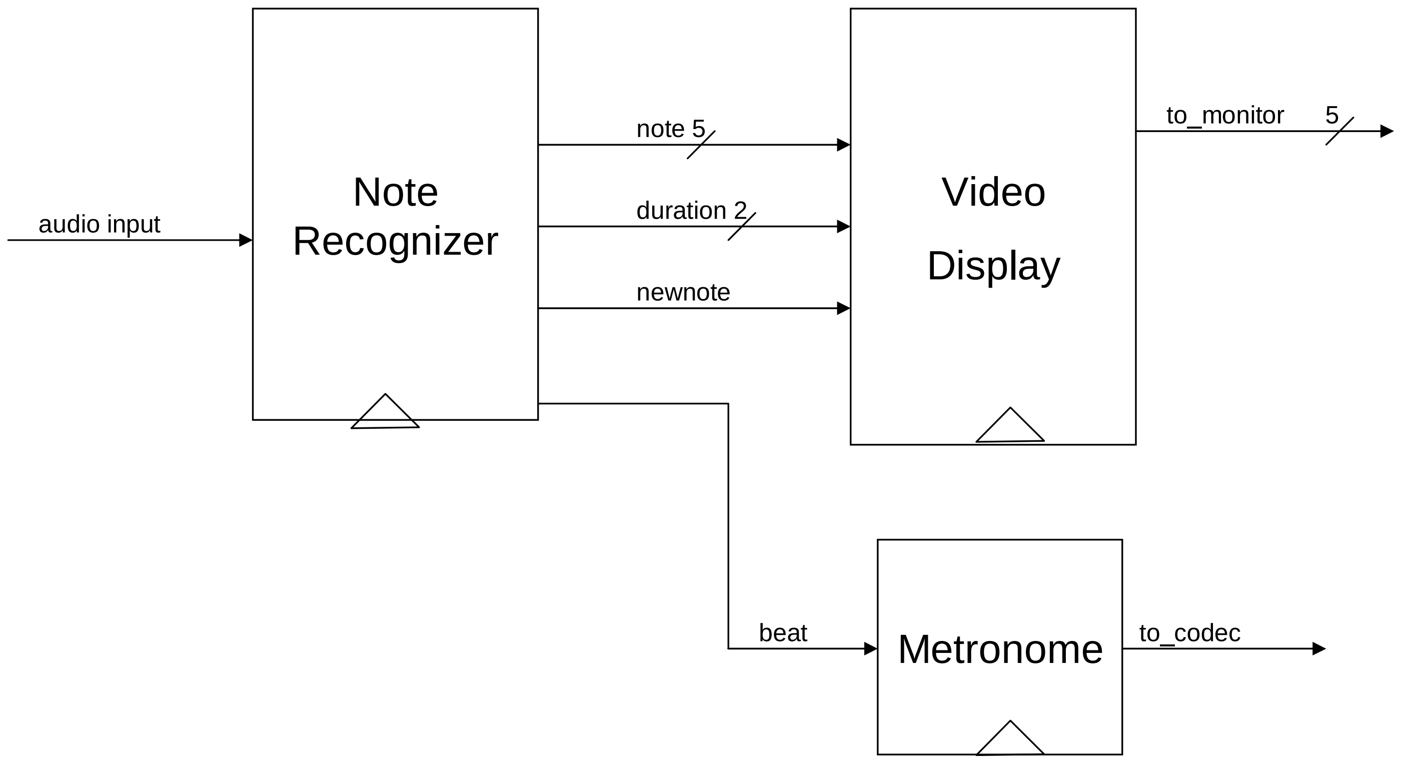
XVGA: The XVGA module will take the *clock* input and generate the appropriate signals for video display *hcount*, *vcount*, *hsync* and *vsync*. It will work for a resolution of 1024x768 and a refresh rate of 60Hz. Finally, it will take the *pixel* input from the Video module in order to generate the image.

ROM: The ROM will store bit values representing the image (pixels ON and pixels OFF) of notes of different duration, which can be requested by inputting the appropriate address (the *duration* signal itself will work). The notes will be approximately 10x20 pixels large. They will be drawn in such a way that they will all align correctly on a stave with respect to their upper-left pixel, so that with same coordinates *cx cy*, every note will be drawn on the same line of the stave.

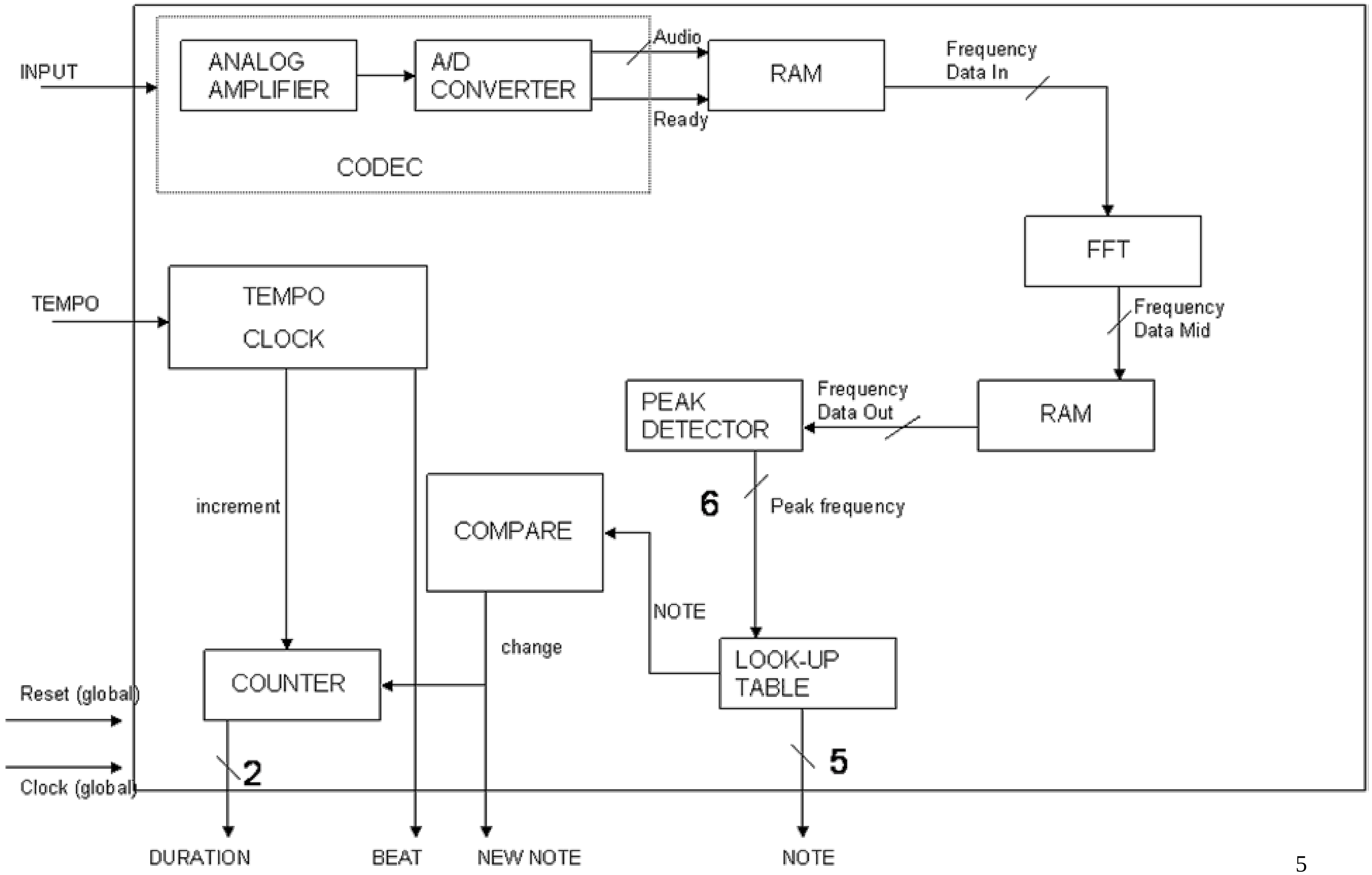
Video: The core of the module. Video will take in the coordinates *cx* and *cy* of the note to display from the tracker, the *color* for the display, the *duration* of the note and the appropriate video signals, and will generate the *pixel* outputs for the screen. At reset, Video will rely on two sprite submodules which will generate pixels for the stave and the treble clefs. When a start signal is given from the Tracker, Video will ask the ROM to generate the appropriate note figure, and will feed it to a logic module. The logic will generate pixels for that figure having coordinates *cx*, *cy* as reference point for the *hcount* and *vcount* of the upper-left pixel. Finally, Video will OR the various pixel signals so as to emit a positive *pixel* (color specified by the color input) if any submodule (stave, clef, note) emits a positive pixel.

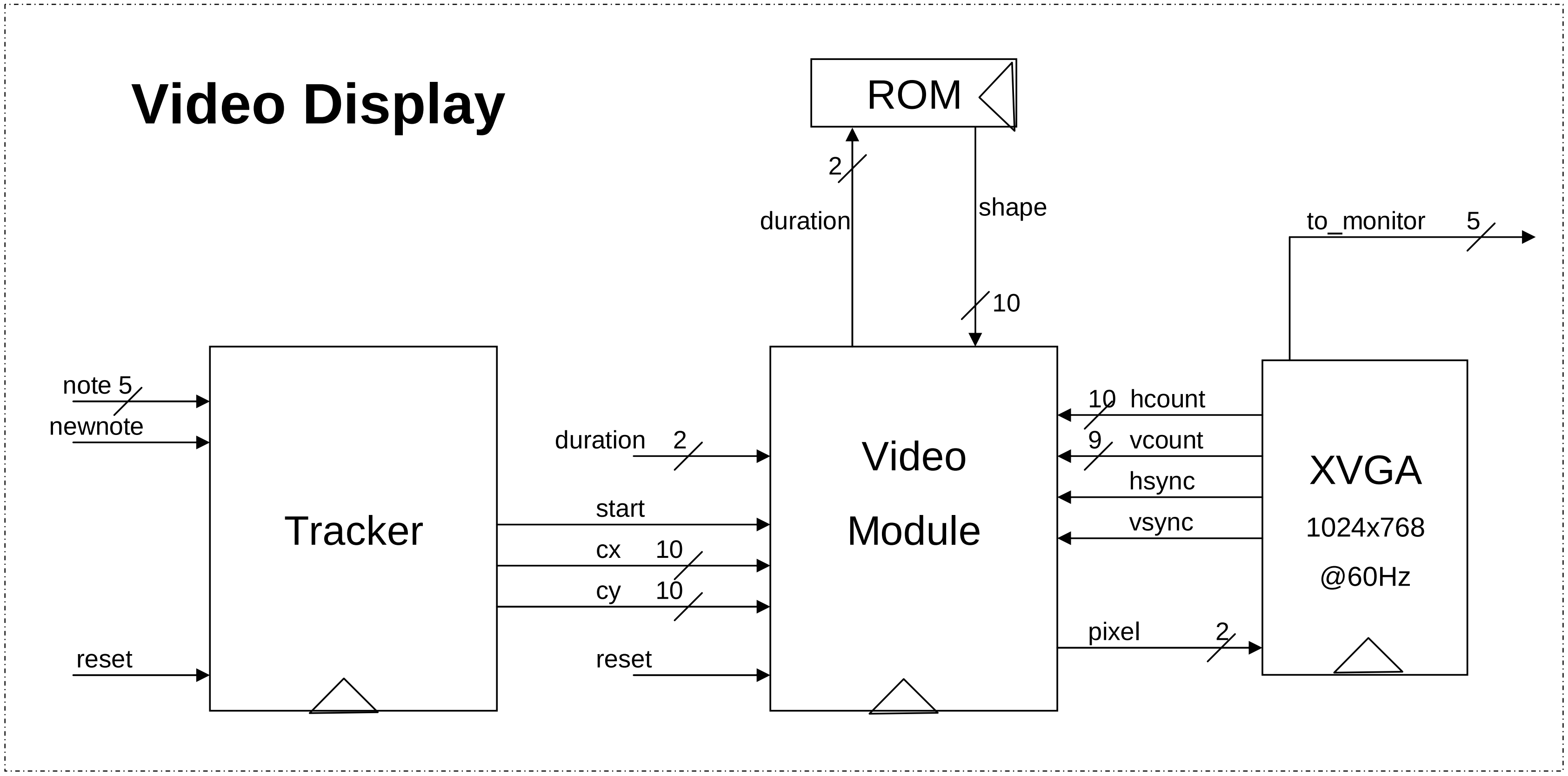
Metronome (Roberto Carli)
Finally, there will be a metronome which will emit a “click” audio signal, so as to let the musician hear the tempo he specified. The Metronome will take as input the signal *beat*, a pulse signal given by the note recognizer representing the tempo of the music, and it will output a “click” audio signal *to_codec* which will go to the audio codec and subsequently to speakers.

Overall Block Diagram



NOTE RECOGNIZER




Video Display
ROM
2
duration
shape
10
to_monitor
5
note 5
newnote
reset
Tracker
duration
2
start
cx
10
cy
10
reset
Video
Module
10 hcount
9 vcount
hsync
vsync
pixel
2
XVGA
1024x768
@60Hz