6.111 Final project checklist
FPGA side-scrolling videogame
Telmo Luis Correa Junior

Modules:

Microprocessor: modified beta; new instruction stalls the processor and sends control signal to external hardware
  Can execute regular beta/assembly code
  Can react properly to the interrupt requests from GPU
  Can react properly to interrupt requests from user input
If time permits:
  Can be further pipelined (more than 2 stages)

GPU: graphics processing unit, responsible for sprite management, VGA signal and collisions.
Components:

Blob manager
  Can produce the appropriate control signals given CPU control signal and exposed registers

Blob
  Can react properly to all control signals from the blob manager
  Can communicate properly with sprite loader to load sprites locally
  Can produce the appropriate pixel information output for every X,Y

Sprite loader
  Can load sprites from appropriate location on sprite ROM
  Can service the blobs with sprites, in series appropriately
  Can inform interrupt generator about GPU busy/ready IRQs

Pixel selection tree
  Each level of the tree can select a pixel appropriately, according to layer, clip bit, and owner
  Each level of the tree can determine generic collision and collision with sprites with “enemy bit” set appropriately
  Collision OR logic can determine collision status for every blob, receiving information from every relevant level on the tree
  Can use background default pixel from blob manager

VGA generator
  Can request X,Y coordinates in series from blob manager and produce the corresponding VGA signal

Interrupt request generator
Can detect changes on collision OR output and generate interrupts
Can generate GPU ready / GPU busy signals based on sprite loader output

If time permits:
Audio processing unit: responsible for producing audio for the game

Audio manager:
  Can receive commands from the exposed CPU registers and generate the appropriate control signals to other modules
  Can keep track of which audio blob was the least used and use it when requested

Wavetable synth:
  Can react to commands from audio manager for switching, pausing or proceeding the BGM
  Can read the BGM ROM note / instrument information
  Can sample the appropriate instrument wave from Wavetable ROM at the right frequency
  If time permits: can apply an ADSR envelope to generated wave
  Can output the wave to adder

Audio loader:
  Can receiver commands from audio manager about individual audio blobs
  Can load wave information from the sound effects ROM
  Can send a new intensity to target audio blob

Audio blob:
  Can keep the received audio signal from audio loader
  Can send audio signal to adder

Adder:
  Can add the 4 received waveforms and send it to the AC97.

Software
  2D program, generates sprites that move on screen depending on user input and sprite collision
  If time permits: side-scrolling platform game level
  If time permits: RAM-based load and save state