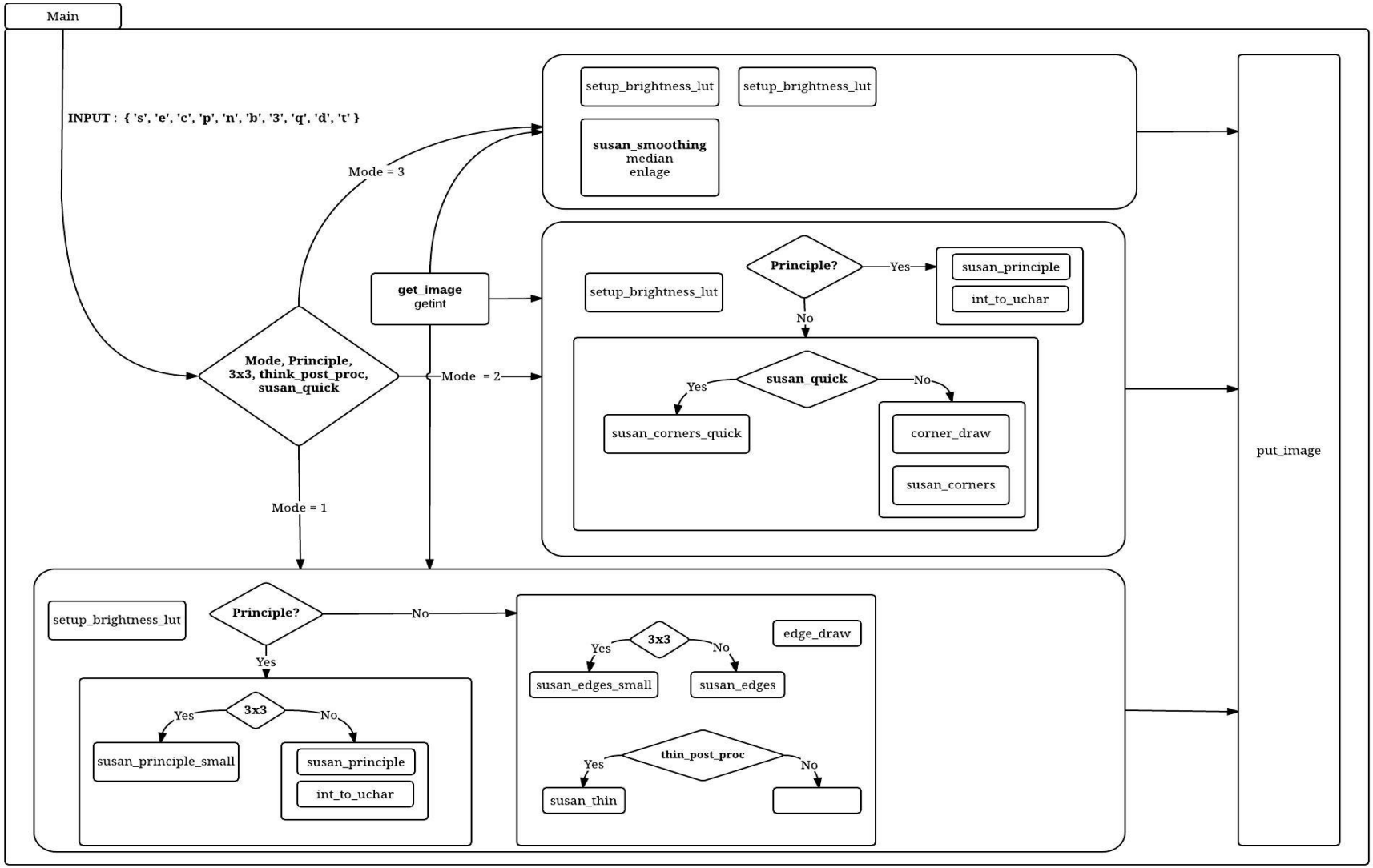
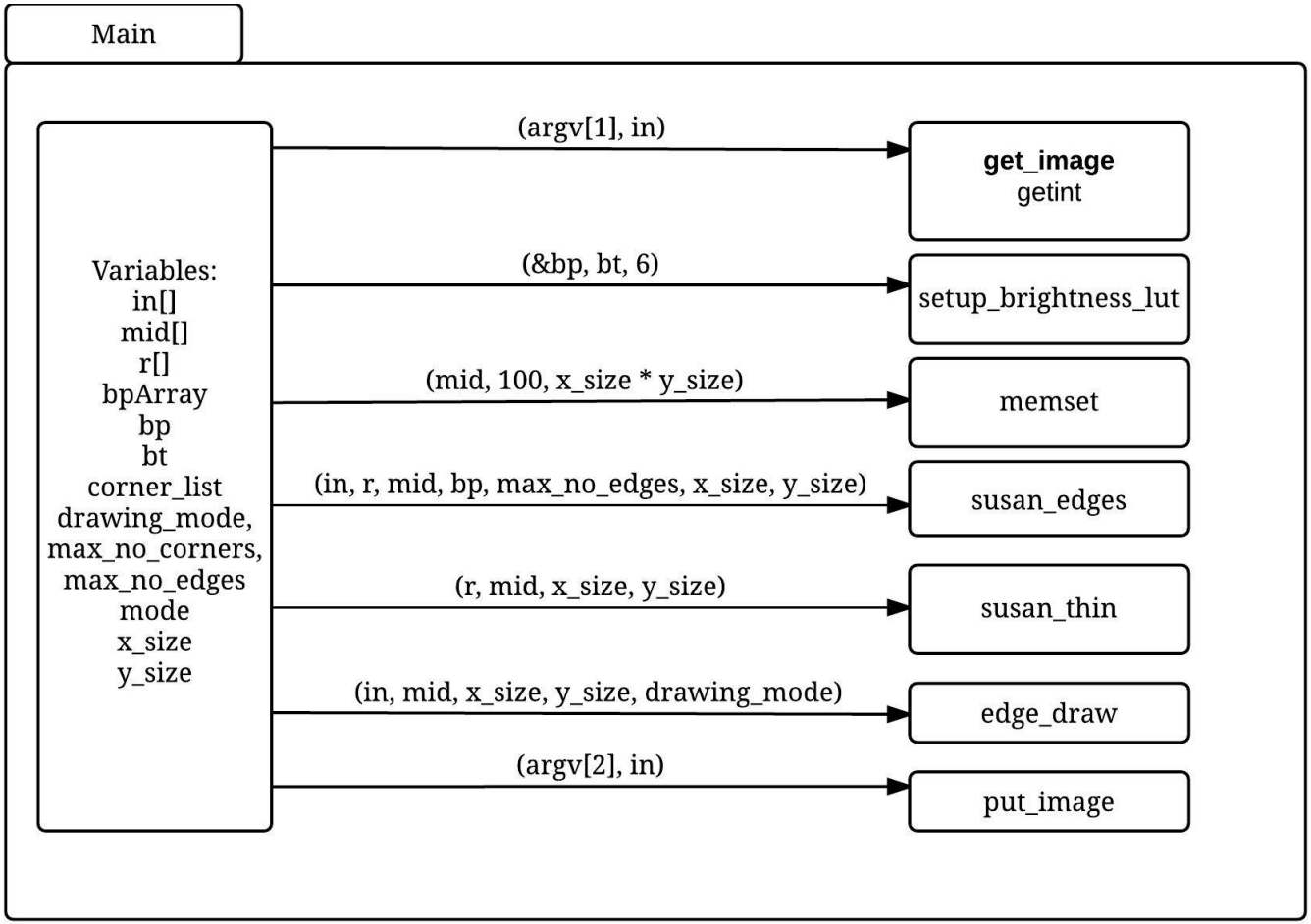
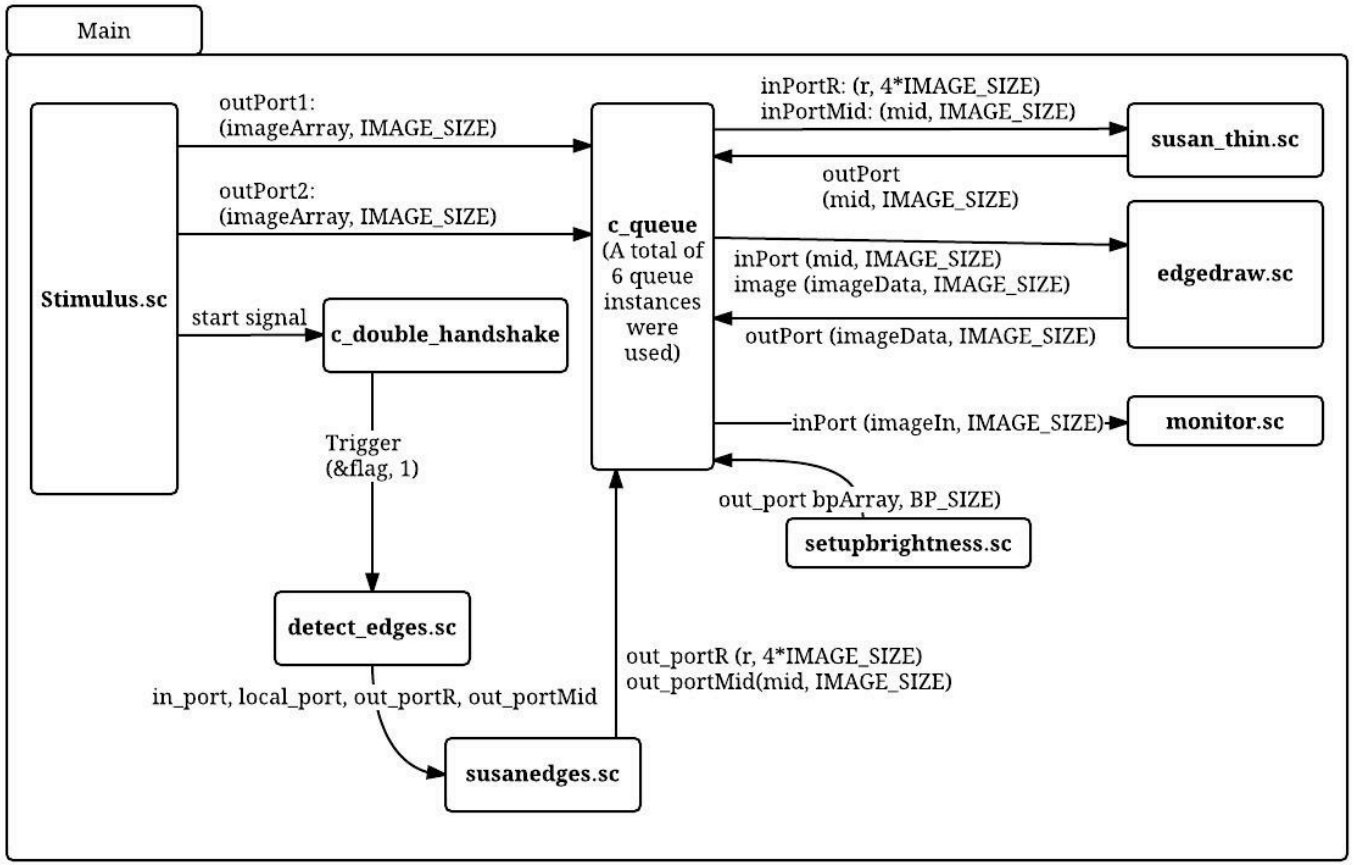
EE382N : Embedded System Design and Modeling
Lab #1
Names: Behzad Boroujerdian, Kishore Punniyamurthy, Keum San Chun
Reference 1]Block diagram of the original version of susan.c



EE382N : Embedded System Design and Modeling
Lab #1
Names: Behzad Boroujerdian, Kishore Punniyamurthy, Keum San Chun
[Reference 2]Block diagram of the simplified version of susan.c



EE382N : Embedded System Design and Modeling
Lab #1
Names: Behzad Boroujerdian, Kishore Punniyamurthy, Keum San Chun
[Reference 3]Block diagram of main.sc



Part C.3

Memory Boundedness

Our model can run in bounded memory provided, each behavior runs at similar rate, if not then, it cannot run in bounded memory. For e.g, stimulus runs at a very fast rate but susan runs very slow, then as time progress, the size of queue grows larger and larger (no bound) . The model can execute with queue size = image size. Limiting the queue size affects the scheduling by bring an order (corresponding to data flow) to execution.

Queue size dependency

Depending on the queue size between behaviors, we can utilize parallelism.We consider three scenarios for queue size:If the size of the queue between behaviors is greater than the imageSize, then connected behaviors can run in parallel. However, if the queue size is equal to the imageSize, the flow will be done in sequential manner as shown below:

Stimulus → read_image → susan → write_image → monitor

If the queue size is less than the imageSize, the program will not work (deadlock)

Queue size Justification:

As long as all the behaviors are working as they are meant to be, and queue size is at least 1 image size, the model is deadlock free and it is deterministic, as there is no shared resource.

D.1

We achieve the following timing number from sequential and parallel running(using –par) of our code:

sequential: 0.45user 1.19system 0:01.80elapsed 91%CPU (0avgtext+0avgdata 115200maxresident
parallell: 0.45user 1.13system 0:01.75elapsed 90%CPU (0avgtext+0avgdata 115728maxresident)
As can be seen, we see a slight improvement on the system time when run in parallel. We believe that this is due to the fact that running our code in multicore, we are capable of running the behaviors in parallel

Part D.2

We chose to parallelize edge draw behavior. In our implementation , we are instantiating 2 instances of edge_draw behavior which processes half image parallely. Edge_draw requires 2 inputs – mid and input_image. We created separate behavior for splitting the mid and image into 2

sections which can be sent each of the edge_draw instance. Within edge_draw, processing each image element modifies few image elements before and after the specific element in consideration. We split the image into 2 parts with overlapping sections, the image is 7220 (IMG_SIZE) elements long, 1 part is image [0: (IMG_SIZE/2 + offset)] and 2nd part is image [(IMG_SIZE/2 – offset) : IMG_SIZE-1], where offset is 133 (decided based on input and dependencies. The processed outputs from each edge_draw instance is then merged together using another newly created behavior. (please refer to the png pictures provided)

This model implemented as KPN reveals the task level parallelism but the parallelism within a task is not explicitly highlighted. There is potentially parallelism to be exploited in susan_thin and susan_Edge behavior which is unknown until implemented parallelly (or expressed in a different computational model where the tool can be guided to synthesize the design in a parallel fashion)
Implementing it using different MOC (like Data flow graph with finer granularity) would show the parallelism within task more explicitly.

Yes, this model(our code) can be expressed using SDF

could a tool be developed to recognize if a model is KPN or SDF ?

- if a program terminates without deadlocking we can claim it is definitely not a KPN(since KPN termination point is an artificial deadlock)
- if there exists a loop in the model, then it is definitely not a KPN.

Advantage KPN over SDF:

Turing complete

Deterministic

Advantage SDF over KPN:

Memory bounded analysis possible

Static schedule of actors

Deadlock analysis

EE382N : Embedded System Design and Modeling
Lab #1
Names: Behzad Boroujerdian, Kishore Punniyamurthy, Keum San Chun

Note: all of our images (regarding the questions above) are stored in a folder called “images”