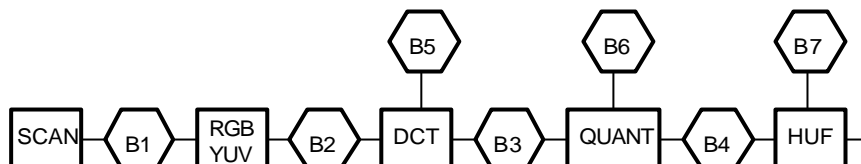


### SYSTEM LEVEL MODELING OF THE JPEG 422 BASELINE ENCODER

---

Now that we have figured out the flow of information in the JPEG encoder, we can go one step further and construct an executable system level model. While we already have a reference implementation in C++, this still leaves a number of issues open. Most important, the current model is sequential and (potential) parallelism is not expressed. When considering hardware or multi-processor software targets, this is vital information.



In order to think about those problems, we will convert the JPEG C++ reference code to an uncommitted system level model. This is a model that can be targeted to as many different implementation styles as possible. Considering the flow-graph in the figure this means that we will introduce concurrency among all the blocks between buffers. Thus, SCAN, RGBYUV, DCT, QUANT and HUF will each become separate processes in a system level model that can express parallelism. The buffers B1 through B7 become shared memory between the processes. Because those buffers are accessed by parallel processes, you will have to make sure that no data precedences are violated. This is done by using a system level communication mechanism (channel or equivalent) that ensures proper synchronization between reader and writer.

We will consider three different environments for constructing the system level model: SystemC, SpecC and HandleC. You will actually need to implement only one – the project homepage indicates which environment should be used by your team. When you have completed this homework, you will have made a partitioned model, where each block (RGBYUV to HUF) can be considered separately for implementation. This makes optimization and refinement tasks toward implementation easier.

### THE ASSIGNMENT

Homework 2 has learned you where the different parts of the JPEG encoder are located in the reference C++ source code. The goal now is to dissect this reference code into different components and describe them in terms of a system level language: HandleC, SystemC or SpecC. You need NOT to write code from scratch! Try to reuse as much of the existing code as possible. You need to construct a system level model that implements the figure above, where each of the processes (SCAN, RGBYUV, DCT, QUANT, HUF) are parallel processes (or tasks).

In order to validate your model, you have to show that your system model generates the same output as the reference model in C++. This means that, given a PPM file, both of them produce an identical JPEG file.

### GETTING STARTED

Check the project homepage to find which target environment (SystemC, SpecC, HandleC) was assigned to you. The very first thing you should do is to setup the environment and try a small example. All of the environments come with prepackaged examples, so take a look at them.

When you develop your model, work incrementally. Don't try to get the whole thing at work at once after some nights of coding – this will only buy you some extra nights of debugging! The easiest way is likely to start at the front, with the SCAN and DCT block, and work your way to the back of the model. Each time you extend the SystemC/SpecC/HandleC model, check it with the reference model in C++ to see if it still does what you want. You can put extra print/cout statements in the C++ code to probe more variables if you want.

Because of the complexity of this task, the only measure of success for this homework is CORRECTNESS. This means that simulation speed, and code size will not be evaluation criteria.

Good Luck.

## WHAT TO TURN IN

Due Date: **5/2/2002** at 12 o'clock noon at Letty's desk (7440 Boelter Hall).

What: Up to 2 sheets containing the following:

- Your name, and environment used
- Link to your homepage where you make the HandleC/SystemC/SpecC model available. Do not provide public access to this page before 5/2/2002 at 12 o'clock
- An answer to the following questions:
  - A **list of specific HandleC/SystemC/SpecC language features** that you have used to solve the problem. An example of a language feature is a specific keyword, operator or type that are only available in that language.
  - Indicate **three clear advantages** of HandleC/SystemC/SpecC that are not offered by C++. Please be specific to the JPEG model, and formulate your answer from the designer point-of-view. It can help to answer this by asking yourself: Do I have a better understanding of my system model than from the C++? If yes, why?
  - Indicate **one clear disadvantage** (if any) of HandleC/SystemC/SpecC. What did you have to give up in the C++ model that was not offered by the new environments?