FSM Hierarchy:

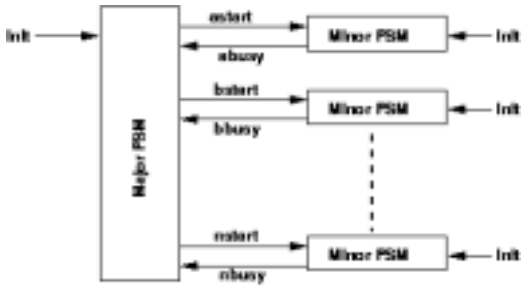
We want to build a control system using multiple FSM's

Minor FSM's are controlled (supervised) by a Major FSM

All FSM's use the same clock

Minor FSM's may be decomposed into multiple FSM's themselves

All FSM's (major and minor) initialized by the same init signal



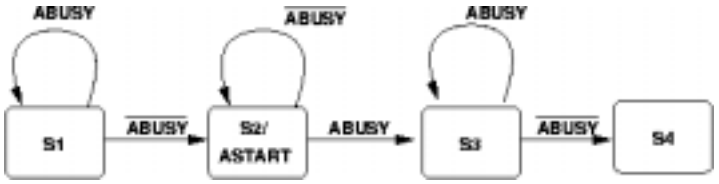
Major FSM invokes minor FSM's using a protocol that is much like a handshake:

Wait until minor FSM is not busy

Invoke minor FSM

Wait while minor FSM does its thing and is not busy again

Go on to the next thing…



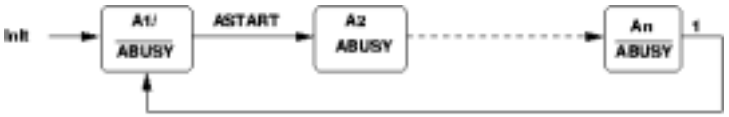
It looks this way from the minor fsm

On init this fsm is not busy

It becomes busy when it is started

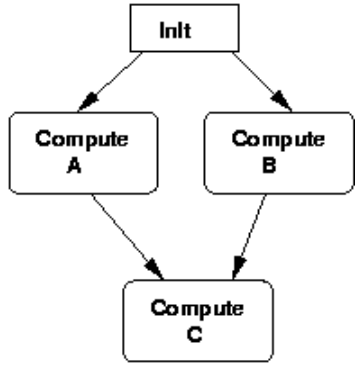
It becomes not busy when it finishes

and goes back to the init state

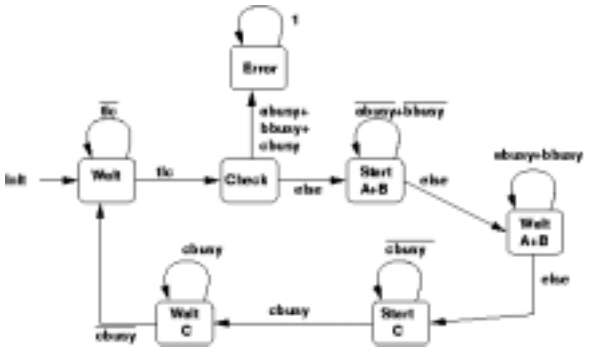


Suppose we want to do two computations in parallel and then a third that depends on the two.

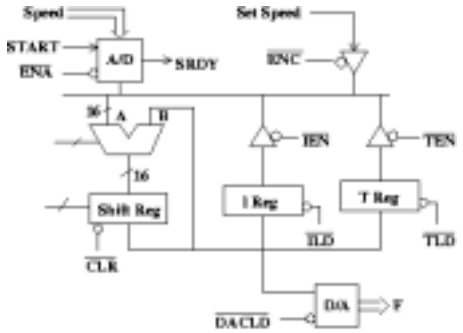
And suppose this is part of a repetitive task, set off by a timer tick.



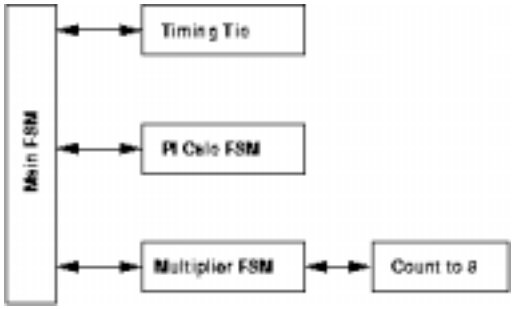
It might look like this: something is wrong if A,B and C are not ready at the timer tic



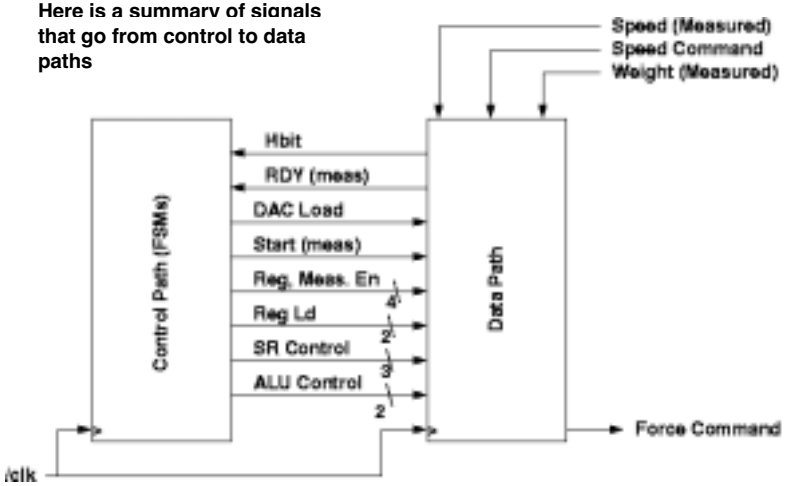
Here is the data path from last time



We could control this with one large FSM, but it seems reasonable to break the control down to several smaller (more easily developed and tested) FSM's, which must then be coordinated



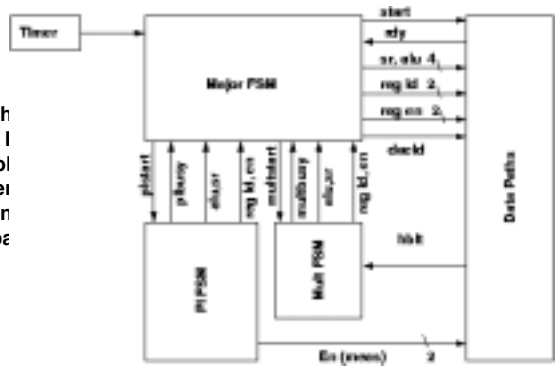
Here is a summary of signals that go from control to data paths



The Main FSM (in this rendition)

a: Controls the process and minor FSMs

b. Multiplexes signals the multiple minor FSMs use



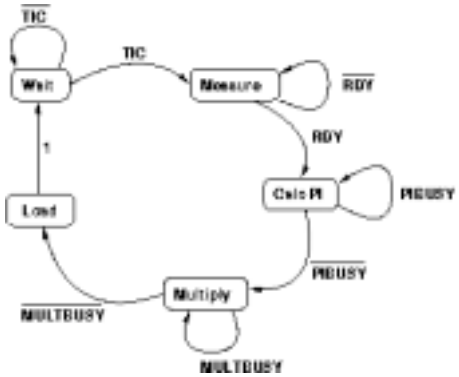
Note th minor I control differer elemen data pa

This is the Major FSM loop for the example of last time

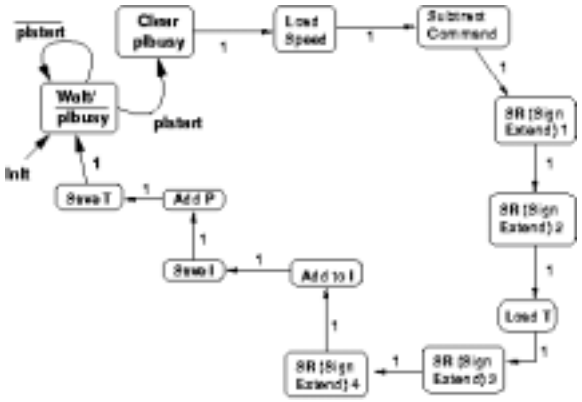
We do not have an error check here.

A: probably do not want the trolley to hang up

B: We are fairly sure we have time for the computations



Here is the minor FSM that controls the PI part of computation



This is the minor fsm that controls the multiplication process. It controls another fsm which is a count zero to seven. No handshake is required for that counter because it is known to take only one clock cycle. The test variable CT is set when the counter reaches 7. The last bit is to rotate the answer back into place.

Figure 1. FLEX 10K Device Block Diagram

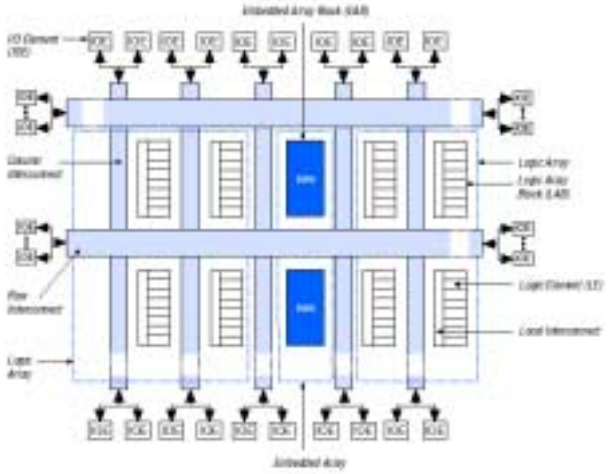


Figure 4. FLEX 10K Embedded Array Block

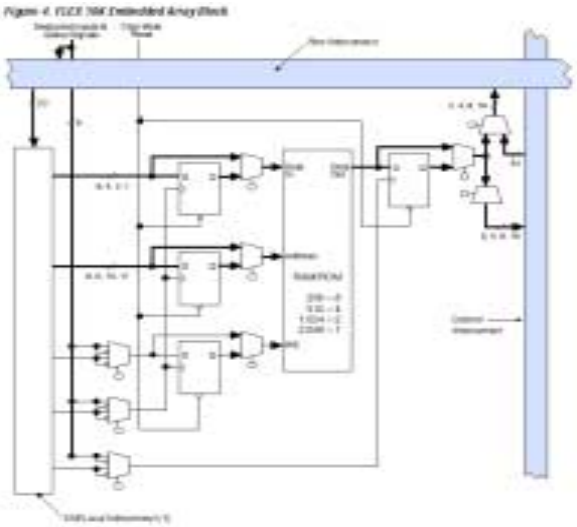


Figure 5. FLEX 10K LAB

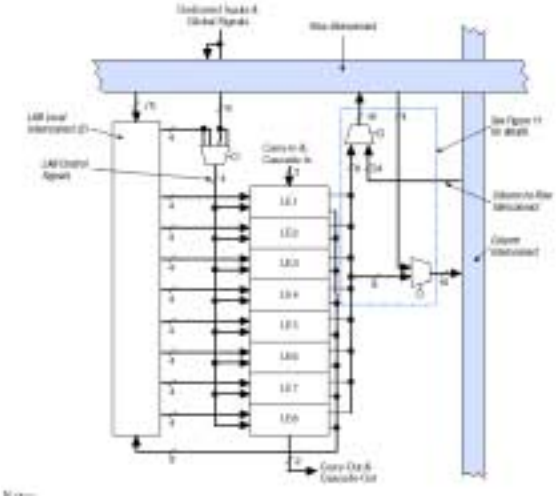


Figure 6. FLEX 10K Logic Element