FPGAs are used all the time in computing these days. Here we examine some fundamental reasons why FPGAs are great for computing.

Why-use-FPGAs-for-Computing

A traditional state machine produces the same output state vector at the same address. At compile time it can have 2^n*Lpossible states where n is the number of output states and L is the number of lines of microcode.

At runtime, the traditional state machine has L different state vectors.

Hotstate Outputs
To control state outputs stimulatingly use the comma operator.
LED0 = 1, LED2 = 0; // This is 100% C
These states will toggle at the same time leaving all other states quiescent.
The state outputs are qualified by the corresponding mask bit and latched if appropriate.
The mask bit and state bits are combined into the state output.
state[i] = mask[i]?new_state[i]:old_state;

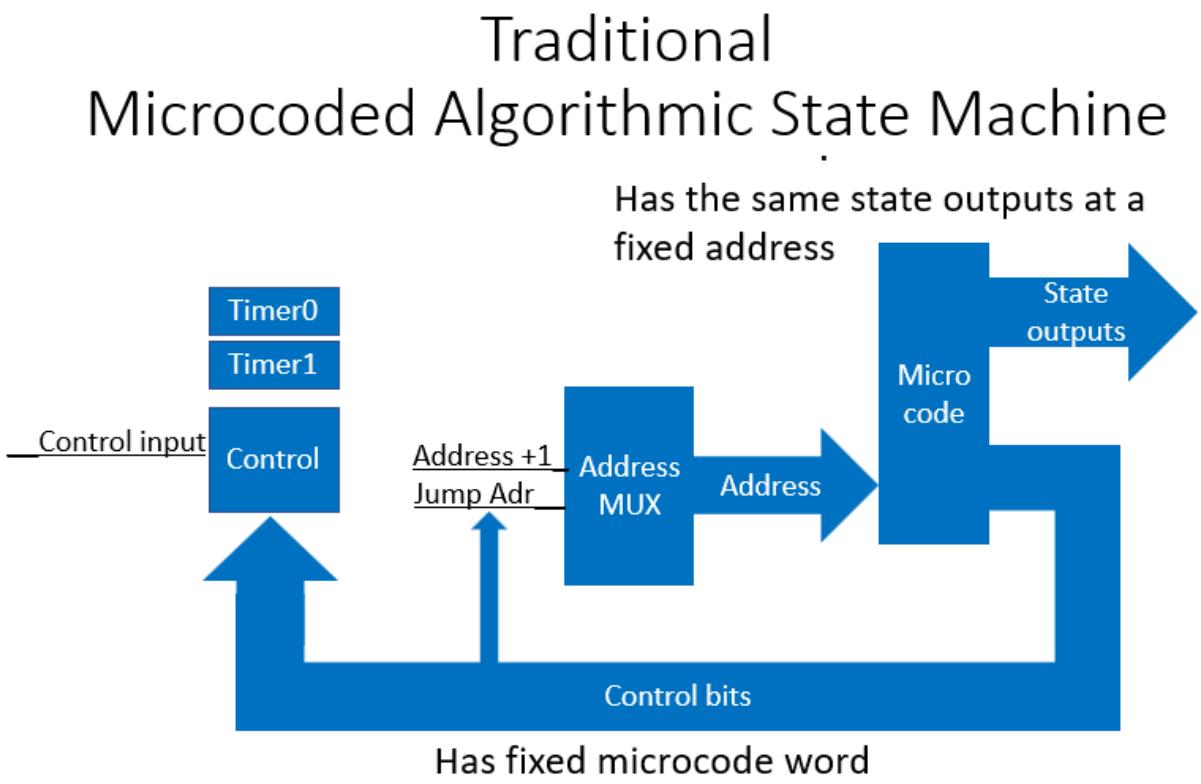
At compile time the hotstate machines have 2^n*L possible states where n is the number of output states and L is the number of lines of microcode.

At runtime the possible number of state outputs at any one address during run time is

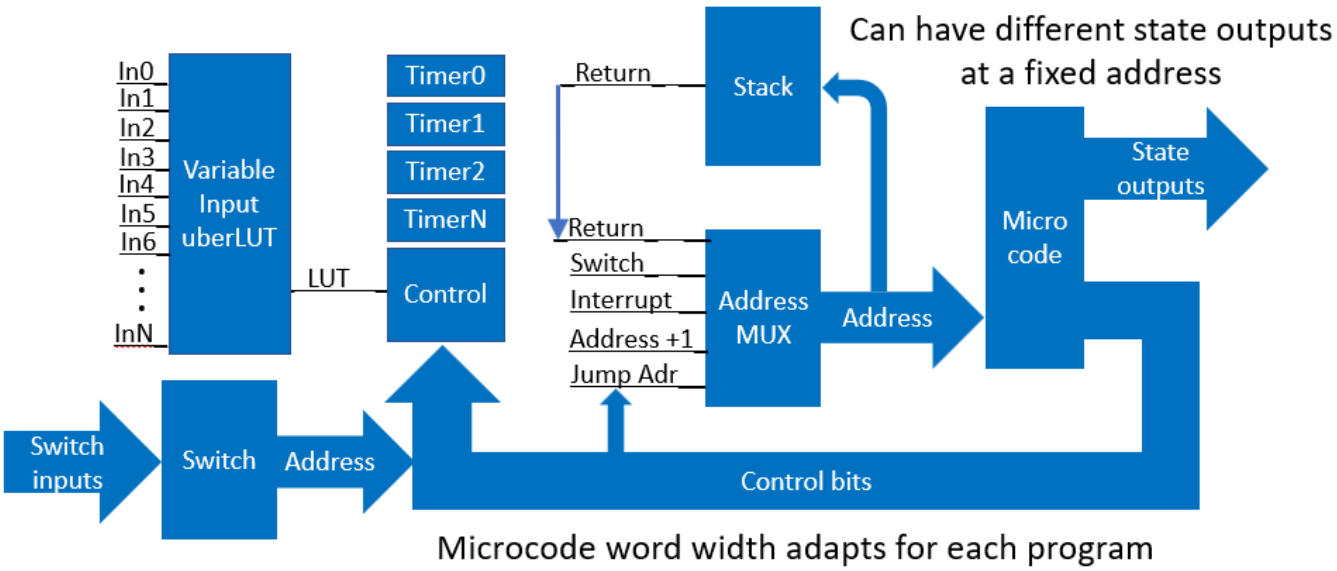
$$\sum_{k=1}^{L} k * 2^{(n-m)}$$

where n is the total number of states and m is the number of states used in that line of code and L is the total number of lines of microcode.

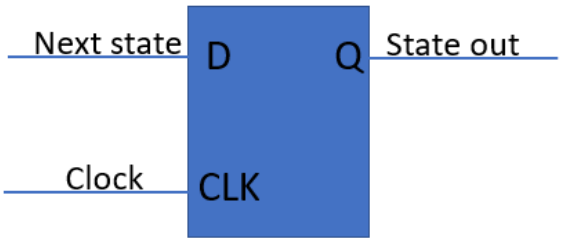
In the traditional algorithmic state, there is one (1) state vector per address.

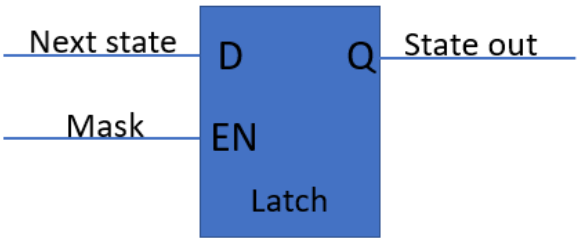


Hotstate Runtime Loadable Microcoded Algorithmic State Machine



The difference in state outputs between traditional and Hotstate state machines





The traditional state machine always passes next state to state out.

At runtime there is one output vector at any one address

Passes next state to state out if mask = 1

At runtime, there are 2^(m-n) states where m is the number of states and n is the number of states used at that address