The Hotstate Machine

By
Steve Casselman, CEO
HotWright Inc.
11/25/2022

The Algorithmic planes

- Algorithms are recipes.
  - A recipe has a list of ingredients
  - And a description of what to do with them
- The ingredients are data of different kinds
  - This is called the data plane
- The description of what to do with the data is the called control plane

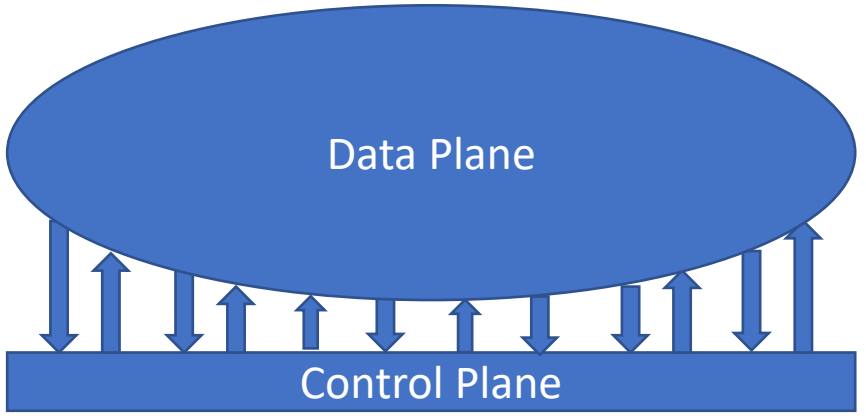
- All algorithms can be fully characterized by their data and control planes

Processors

- Processors are fixed pieces of hardware designed to run any program
- The data plane is implemented in an arithmetic-logic unit (ALU) and the register file.
- The control plane is implemented in the sequencer, instruction decoder and algorithmic state machine (ASM)
- Most processor ASMs are microcoded ASMs that allow for improvements/fixes to ASM.

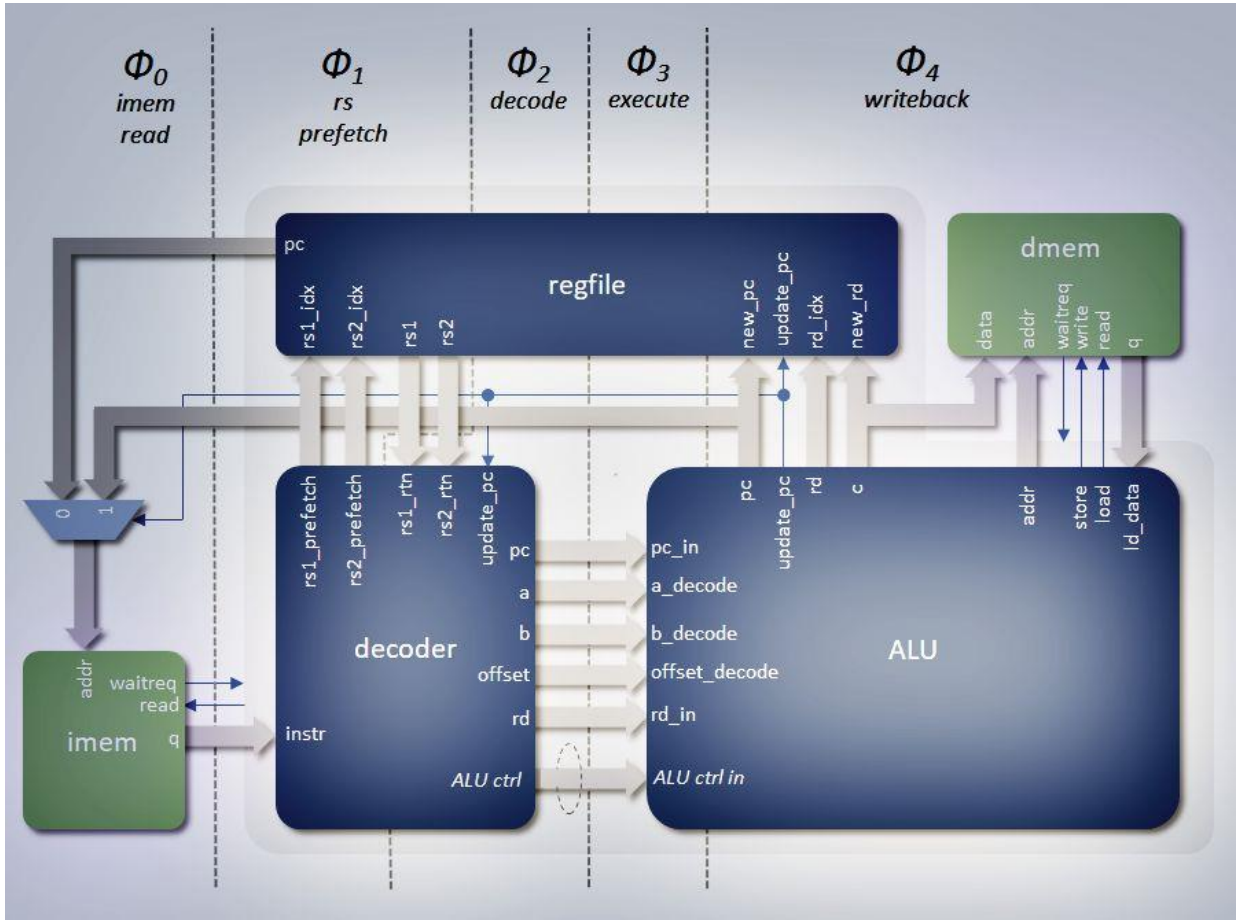
Reconfigurable computing

- An algorithm can be decomposed into a data plane and a control plane
- The data plane is a cloud of logic, arithmetic, and registers
- The control plane manages data movement and monitors status of the data plane



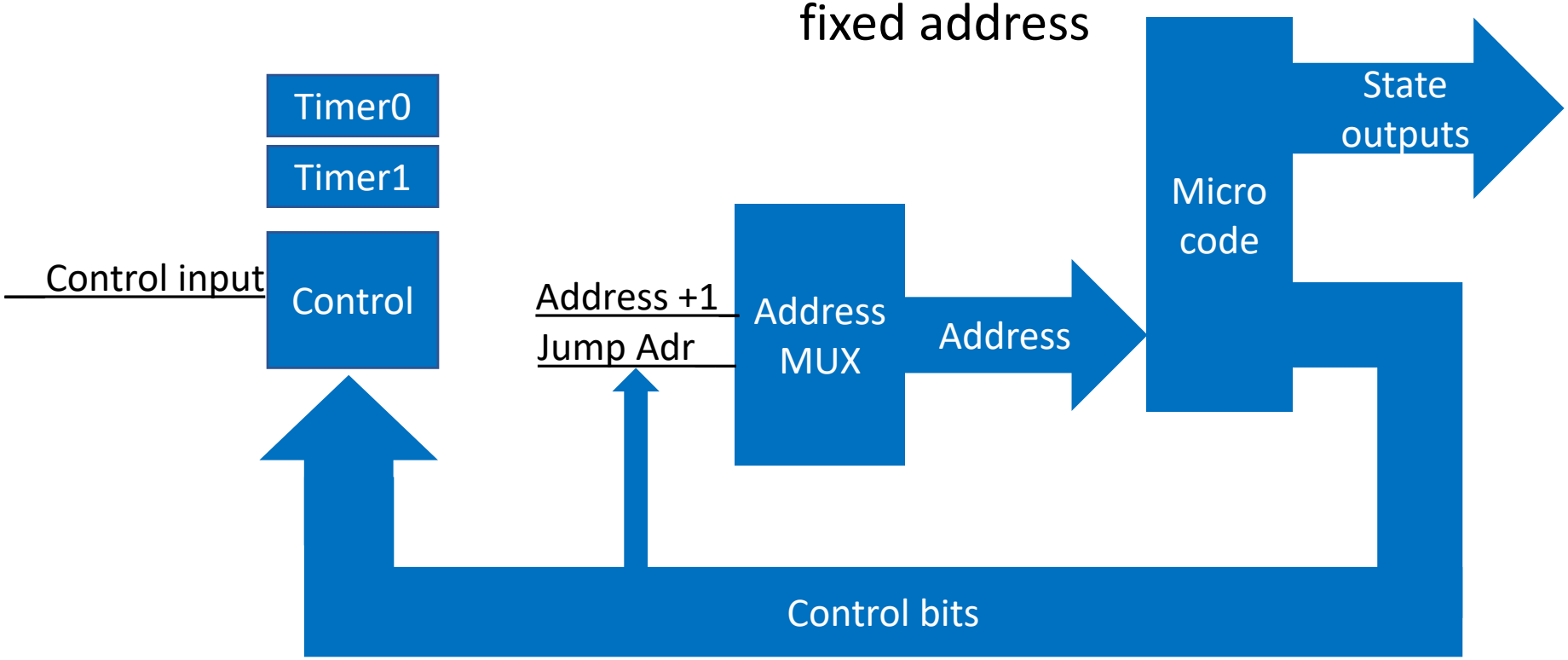
Block Diagram for standard data processor

- The decoder block oversees everything. It takes in instructions and orchestrates all the data movement.



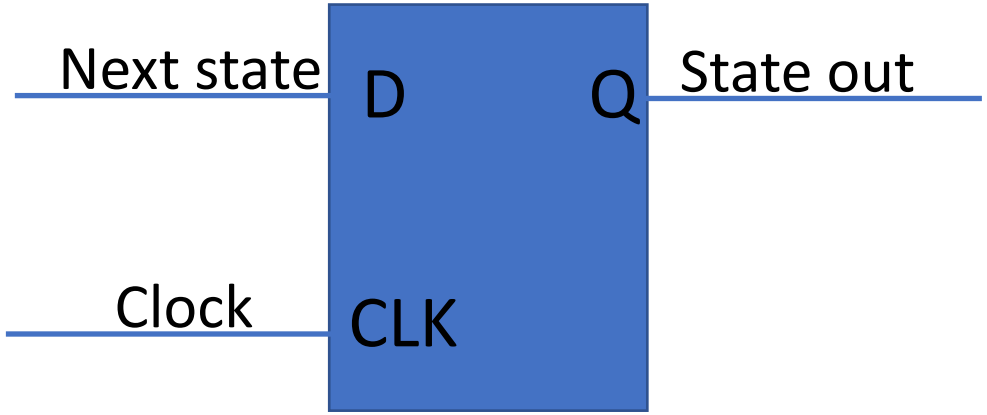
Traditional Microcoded Algorithmic State Machine The decoder block in the previous slide

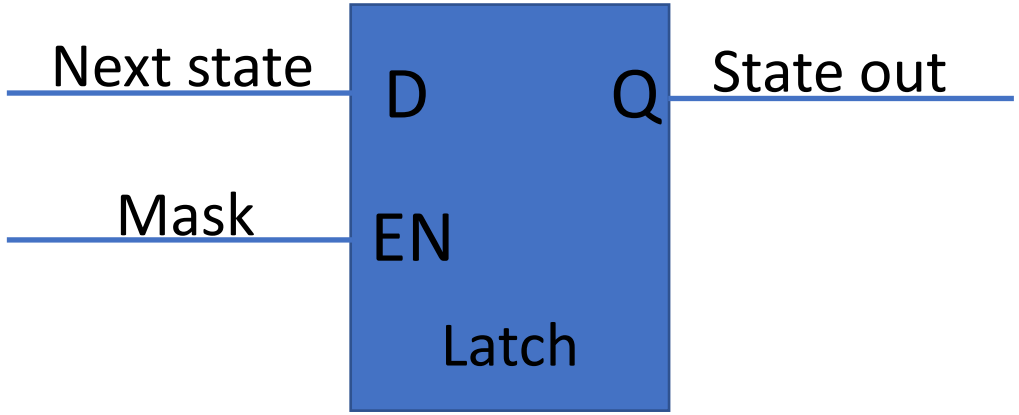
Has the same state outputs at a fixed address



Has fixed microcode word

The difference in state outputs between traditional and the Hotstate machine





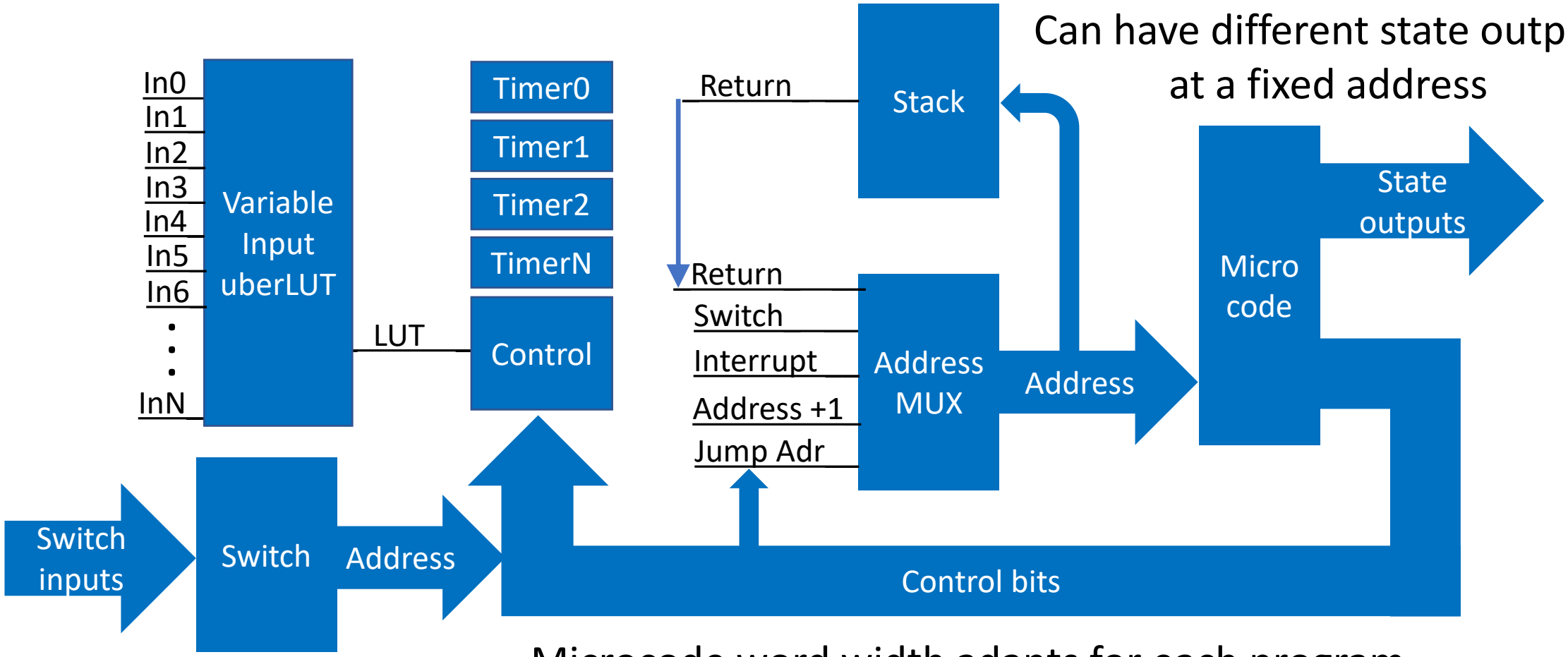
The traditional state machine always passes next state to state out.

At runtime, there is one output vector at any one address

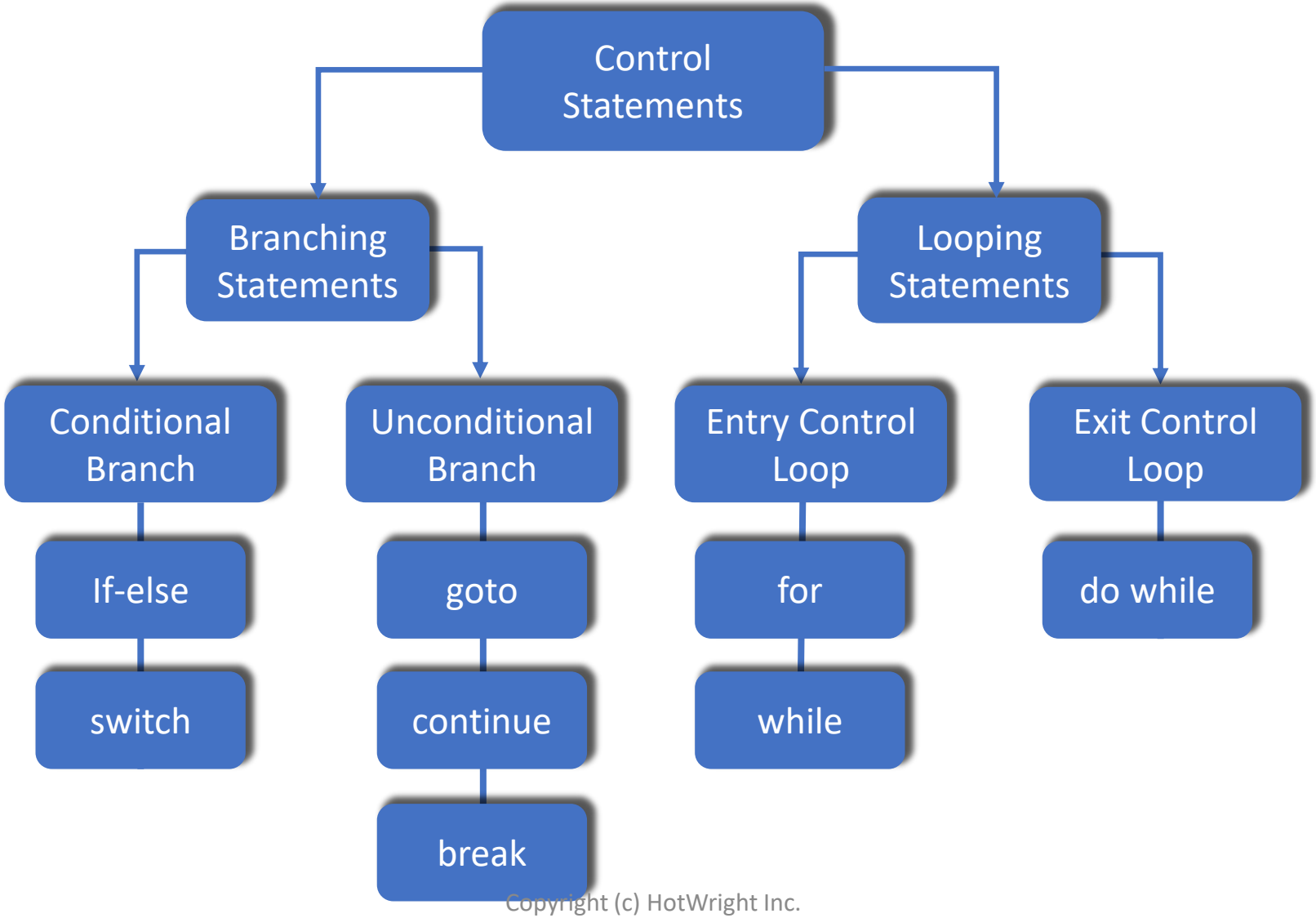
Passes next state to state out if mask = 1

At runtime, there are $2^{(n-m)}$ states where n is the number of states and m is the number of states at any one address

Hotstate Runtime Loadable Microcoded Algorithmic State Machine



Hotstate C Control Statements



Hotstate Comma C compiler takes program in C99 and a little user Verilog

```
#include "ascii_defs_in.h"
bool read_data= 0;
bool write_data = 0;
bool next = 0;
bool in0;
bool in1;
bool in2;
bool in3;
bool in4;
bool in5;
bool in6;
bool in7;
bool next_wrap_around;
void main() {
  // get rid of non-letters in front
  while (!(lower_case|upper_case)) {
    read_data = 1, next = 1;
    read_data = 0, next = 0;
  }
  // do the rest
  while(1) {
  if ((lower_case|upper_case)) {
      write_data = 1, read_data = 1, next = 0;
      write_data = 0, read_data = 0;
  }
  else if (next_wrap_around == 0) {
   next = 1;
  }
  else {
    read_data = 1;
    read_data = 0;
  }
  }
}
```

Code can be debugged with gdb

```
// This is the user file inclued in parse_tb.v
// You have access to clk, rst, hlt as well as all output states and input variables

integer file, count, outfile,result;
reg [7:0] data_in [0:65535];
reg [7:0] data_out [0:65535];
reg [15:0] data_in_address;
reg [15:0] data_out_address;


initial begin
file = $fopen("text.txt","r");
outfile = $fopen("parse.out","w");
for (count = 0; count < 65536;count = count+1) begin
    data_in[count] = $fgetc(file);
    data_out[count] <= 0;
end

end

reg next_r;

always @(posedge clk) begin
if (ready == 0) begin
count <= 0;
next_r <= 0;
data_in_address <= 0;
data_out_address <= 0;
end
else if (read_data == 1 && next == 0) begin
     data_in_address <= data_in_address + 1;
     if (write_data == 1) begin
          data_out[data_out_address] <= data_in[data_in_address];
          $fwrite(outfile,"%c",data_in[data_in_address]);
          data_out_address <= data_out_address + 1;
     end
end
else if (next == 1 && next_r == 0 && read_data == 0) begin
   data_out[data_out_address] <= 8'ha;
   $fwrite(outfile,"%c",8'h0a);
   data_out_address <= data_out_address + 1;
end
else if (next == 1 && read_data == 1) data_in_address <= data_in_address + 1;

next_r <= next;

if (data_in_address > 65530) begin
   $fclose(outfile);
   $finish();
end
```

```
 State Machine Microcode derived from parse.c

                 s     f
 a               t t   o
 d         v t   s i b r
 d s       a i t t m r c
 r t m   j r m i e / a e
 e a a   a S S m C v n j s r
 s t s   d e e e L a a c m u t
 s e k   r l l l d p r h p b n
-------------------------------
00 0 7 00 0 x x 1 0 0 0 0 0   main()
01 0 0 05 0 x x 0 0 1 0 0 0   while (!((in0 == 1) & (in1 == 0) // over 450 boolean ops per clock
02 5 5 00 0 x x 1 0 0 0 0 0   read_data=1,next=1;
03 0 5 00 0 x x 1 0 0 0 0 0   read_data=0,next=0;
04 0 0 01 0 x x 0 0 0 1 0 0   }
05 0 0 10 1 x x 0 0 1 0 0 0   while (1) {
06 0 0 0a 2 x x 0 0 1 0 0 0   if ((in0 == 1) & (in1 == 0) & ( // over 450 boolean ops per clock
07 3 7 00 0 x x 1 0 0 0 0 0   write_data=1,read_data=1,next=0;
08 0 3 00 0 x x 1 0 0 0 0 0   write_data=0,read_data=0;
09 0 0 0f 0 x x 0 0 0 1 0 0   else
0a 0 0 0d 3 x x 0 0 1 0 0 0   if ((next_wrap_around == 0))
0b 4 4 00 0 x x 1 0 0 0 0 0   next=1;
0c 0 0 0f 0 x x 0 0 0 1 0 0   else
0d 1 1 00 0 x x 1 0 0 0 0 0   read_data=1;
0e 0 1 00 0 x x 1 0 0 0 0 0   read_data=0;
0f 0 0 05 0 x x 0 0 0 1 0 0   }
10 0 0 10 0 x x 0 0 0 1 0 0   :exit

State Assignments
state 0 is read_data
state 1 is write_data
state 2 is next

Variable Assignments
var 0 is in0
var 1 is in1
var 2 is in2
var 3 is in3
var 4 is in4
var 5 is in5
var 6 is in6
var 7 is in7
var 8 is next_wrap_around
```

Output from analyzer

Hotstate runtime loadable parser input and output

The parser program takes the text below and turns it into the text on the right

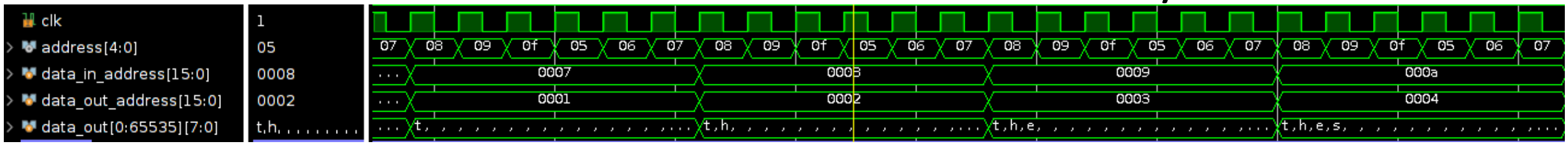
```
.,., these are some words to be parsed.......


"Since conventional computing has reached its limits, new computational methods and devi
ces are being developed," adds Ivan Schuller, a UCSD physicist and one of the paper's au
thors. "These have the potential of revolutionizing computing and in ways that may one d
ay rival the human brain."

In recent years, scientists have sought to make advances in what is known as "neuromorph
ic computing"▒~@~Ta process that seeks to mimic the functionality of the human brain. Be
cause of its human-like characteristics, it may offer more efficient and innovative ways
 to process data using approaches not achievable using existing computational methods.
~
```

```
these
are
some
words
to
be
parsed
Since
conventional
computing
has
reached
its
limits
new
computational
methods
and
devices
are
being
developed
adds
Ivan
Schuller
a
UCSD
physicist
and
one
of
```

Executes one micro instruction every clock



The address is the internal address machine.

Address	Description
5	Start of the while (1) loop.
6	Compares which ascii characters get written into memory
7	Sets write_data = 1 and read_data = 1 and makes sure next = 0
8	Sets write_data and read_data back to zero. One clock between 7 and 8
9	If we don't have upper-case or lower-case character
f	Go back to address 5

The hotstate machine is a versatile algorithmic state machine, programed in a subset of C

- Software complier
  - Extracts parameters from the code to right size hardware
  - Generates the microcode and contents of other memories
  - Generates verilog testbenches
  - Generates a verilog templet for use in an HDL project
  - Generates a makefile
  - Generates a builtin _user() C debugging
  - Generates a user.v for hardware debugging
- Hardware state machine
  - Implements all C control statements plus limited function calls
  - Is highly parameterized and is automatically sized by the compiler
  - The size is different for each program and very small for small programs
  - The machine can be halted or reloaded and reset at runtime
  - Executes one microcode step per clock
  - Can be used as a runtime loadable machine or in standalone mode where the microcode is compiled into the hardware.