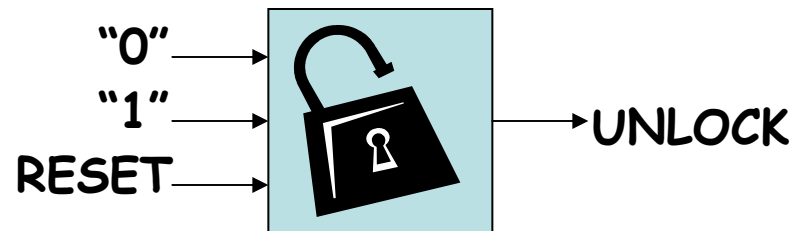


# Demo!

## GOAL:

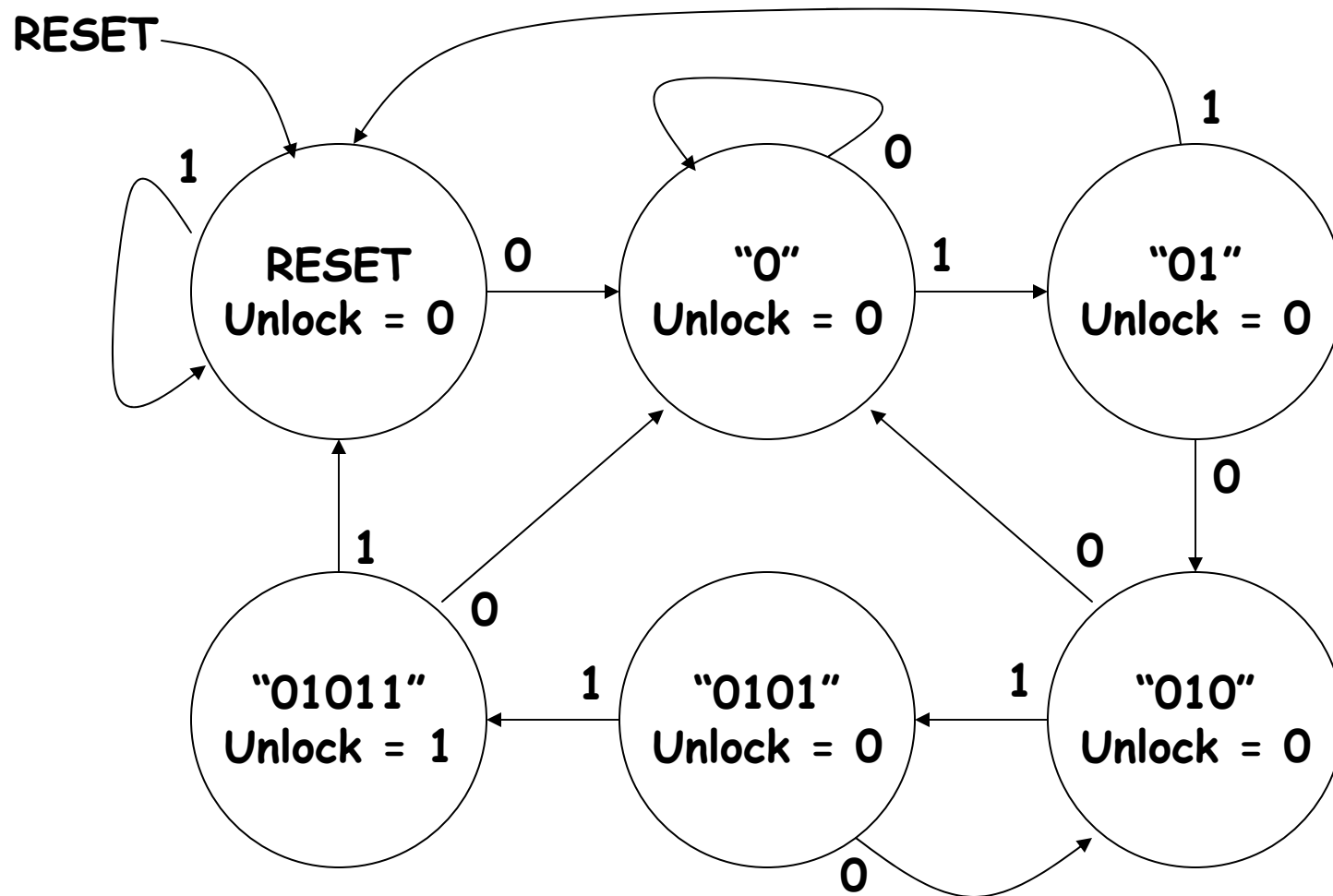
Build an electronic combination lock with a reset button, two number buttons (0 and 1), and an unlock output. The combination should be 01011.



## STEPS:

1. Create state transition diagram for lock FSM
2. Write Verilog module(s) that implement FSM
3. Use MAX+plusII (synthesis, simulation)
4. Program FGPA, wire up buttons, give it a whirl!

# Step 1: State transition diagram



6 states → 3 bits

# Step 2: Write Verilog

```
module lock(clk,reset_in,b0_in,b1_in,out);
  input clk,reset,b0_in,b1_in;
  output out;

  // synchronize push buttons, convert to pulses

  // implement state transition diagram
  reg [2:0] state;
  always @ (posedge clk)
  begin
    state <= ???;
  end

  // generate output
  assign out = ???;

  // debugging?
endmodule
```

# Step 2A: synch buttons

```
// button -- push button synchronizer and level-to-pulse converter
// OUT goes high for one cycle of CLK whenever IN makes a
// low-to-high transition.
```

```
module button(clk,in,out);
```

```
  input clk;
  input in;
  output out;
```

```
  reg r1,r2,r3;
```

```
  always @ (posedge clk)
```

```
  begin
```

```
    r1 <= in;    // first reg in synchronizer
```

```
    r2 <= r1;    // second reg in synchronizer, output is in sync!
```

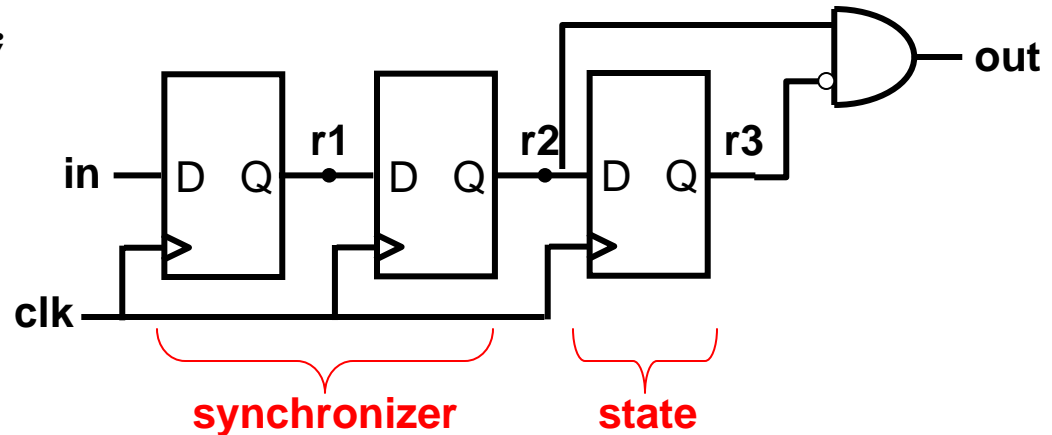
```
    r3 <= r2;    // remembers previous state of button
```

```
  end
```

```
  // rising edge = old value is 0, new value is 1
```

```
  assign out = ~r3 & r2;
```

```
endmodule
```



# Step 2B: state transition diagram

```
// state assignments
parameter S_RESET = 0;
parameter S_0 = 1;
parameter S_01 = 2;
parameter S_010 = 3;
parameter S_0101 = 4;
parameter S_01011 = 5;

// implement state transition diagram
reg [2:0] state;
always @ (posedge clk)
begin
    if (reset) state <= S_RESET;
    else case (state)
        S_RESET: state <= b0 ? S_0      : b1 ? S_RESET : state;
        S_0:      state <= b0 ? S_0      : b1 ? S_01    : state;
        S_01:     state <= b0 ? S_010    : b1 ? S_RESET : state;
        S_010:    state <= b0 ? S_0      : b1 ? S_0101   : state;
        S_0101:   state <= b0 ? S_010    : b1 ? S_01011  : state;
        S_01011: state <= b0 ? S_0      : b1 ? S_RESET : state;
        default: state <= S_RESET; // handle unused states
    endcase
end
```

## Step 2C: generate output

```
// it's a Moore machine!  Output only depends on current state  
  
assign out = (state == S_01011);
```

## Step 2D: debugging?

```
// hmmm.  What would be useful to know?  Current state?  
  
led_decoder l(state,segments);
```

# Step 2E: led\_decoder

```
// led_decoder -- decode 4-bit input into 7-segment enables
module led_decoder(in,segments);
  input [3:0] in;
  output [6:0] segments;
  wire a,b,c,d,e,f,g;    // the seven segments

  // top segment
  assign a = (in==0) || (in==2) || (in==3) || (in==5) || (in==6) || (in==7) || (in==8) ||
             (in==9) || (in==10) || (in==14) ||(in==15);
  // upper right segment
  assign b = (in==0) || (in==1) || (in==2) || (in==3) || (in==4) || (in==7) || (in==8) ||
             (in==9) || (in==10) || (in==13);
  // lower right segment
  assign c = (in==0) || (in==1) || (in==3) || (in==4) || (in==5) || (in==6) || (in==7) ||
             (in==8) || (in==9) || (in==10) || (in==11) || (in==13);
  // bottom segment
  assign d = (in==0) || (in==2) || (in==3) || (in==5) || (in==6) || (in==8) || (in==11) ||
             (in==12) || (in==13) || (in==14);
  // bottom left segment
  assign e = (in==0) || (in==2) || (in==6) || (in==8) || (in==10) || (in==11) || (in==12) ||
             (in==13) || (in==14) || (in==15);
  // upper left segment
  assign f = (in==0) || (in==4) || (in==5) || (in==6) || (in==8) || (in==9) || (in==10) ||
             (in==11) || (in==14) || (in==15);
  // middle segment
  assign g = (in==2) || (in==3) || (in==4) || (in==5) || (in==6) || (in==8) || (in==9) ||
             (in==10) || (in==11) || (in==12) || (in==13) || (in==14) || (in==15);
  // bundle results into a single 7-bit vector
  assign segments = {g,f,e,d,c,b,a};
endmodule
```

# Step 2: final Verilog implementation

```
module lock(clk,reset_in,b0_in,b1_in,out,segments);
    input clk,reset_in,b0_in,b1_in;
    output out;
    output [6:0] segments;

    wire reset,b0,b1;
    button b_reset(clk,reset_in,reset);
    button b_0(clk,b0_in,b0);
    button b_1(clk,b1_in,b1);

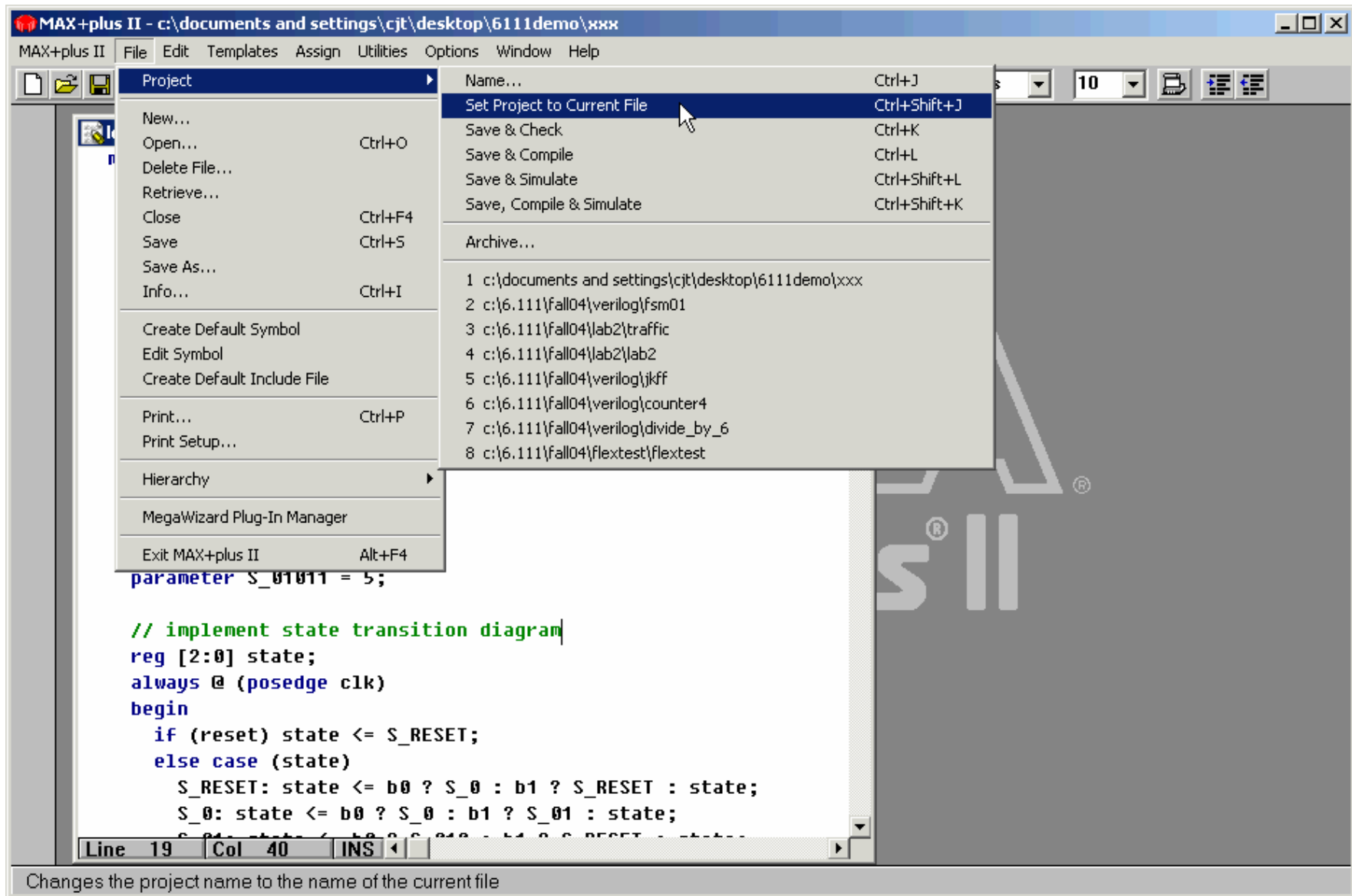
    // state assignments
    parameter S_RESET = 0; parameter S_0 = 1; parameter S_01 = 2;
    parameter S_010 = 3; parameter S_0101 = 4; parameter S_01011 = 5;

    // implement state transition diagram
    reg [2:0] state;
    always @ (posedge clk) begin
        if (reset) state <= S_RESET;
        else case (state)
            S_RESET: state <= b0 ? S_0 : b1 ? S_RESET : state;
            S_0: state <= b0 ? S_0 : b1 ? S_01 : state;
            S_01: state <= b0 ? S_010 : b1 ? S_RESET : state;
            S_010: state <= b0 ? S_0 : b1 ? S_0101 : state;
            S_0101: state <= b0 ? S_010 : b1 ? S_01011 : state;
            S_01011: state <= b0 ? S_0 : b1 ? S_RESET : state;
            default: state <= S_RESET; // handle unused states
        endcase
    end

    assign out = (state == S_01011);
    led_decoder l(state,segments); // debugging!
endmodule
```



# Step 3A: set top level file



# Step 3B: compile/synthesize/map

The screenshot shows the MAX+plus II IDE with a Verilog HDL file named 'lock.v' open in a text editor. The code defines a module 'lock' with inputs 'clk', 'reset\_in', 'b0\_in', and 'b1\_in', and outputs 'out' and 'segments'. The code includes a 'wire' declaration and a 'button' component instantiation. A 'Compiler' dialog box is open, showing various compilation steps: Compiler Netlist Extractor, Database Builder, Logic Synthesizer, Partitioner, Fitter, Timing SNF Extractor, and Assembler. A progress bar is visible below these steps. Below the compiler dialog, a 'Messages - Compiler' window displays an error message: 'Error: Line 6: File c:\documents and settings\cjt\desktop\6111demo\lock.v: Verilog HDL syntax error: Instance name "b0" has already been declared or used'. The error message is highlighted in black. Below the message, there are controls for navigating through messages: 'Message' (1 of 1), 'Locate' (0 of 1), and a 'Locate All' button. A 'Help on Message' button is also present. The status bar at the bottom shows 'Line 4 Col 25 INS'.

```
module lock(clk,reset_in,b0_in,b1_in,out,segments);
input clk,reset_in,b0_in,b1_in;
output out;
output [6:0] segments;

wire reset,b0,b1;
button b_reset(clk,reset_in,reset);
button b0(b0_in,clk,reset);
button b1(b1_in,clk,reset);

// state machine
param mode = 0;
param b0 = 0;
param b1 = 0;
param b2 = 0;
param b3 = 0;
param b4 = 0;

// implementation
reg [2:0] state;
always @(posedge clk)
begin
if (state == 0)
else if (state == 1)
state = 2;
state = 3;
state = 4;
state = 5;
end
```

# Step 3C: start waveform editor

The screenshot shows the MAX+plus II software interface. The main window displays a Verilog code editor with the following code:

```
module lock(  
input clk,  
output out  
output [6:  
  
wire re  
button I  
button I  
button I  
  
// state  
paramete  
paramete  
paramete  
paramete  
paramete  
paramete  
  
// imple  
reg [2:0  
always @  
begin  
if (re  
else @  
S_RI  
S_0: state <= b0 ? S_0 : b1 ? S_01 : state;  
S_01: state <= b0 ? S_01 : b1 ? S_RESET : state;
```

The 'Node' menu is open, and the 'Enter Nodes from SNF...' option is selected. The 'Untitled1 - Waveform Editor' window is also open, showing a waveform with a signal named 'Value' that is high from 0.0ns to 100.0ns. The waveform editor has a time scale of 29.1ns and an interval of 29.1ns. The status bar at the bottom indicates 'Line 8 Col 12 INS'.

MAX+plus II - c:\documents and settings\cjt\desktop\6111demo\lock

MAX+plus II File Edit View Node Assign Utilities Options Window Help

Insert Node... Double-Click  
Enter Nodes from SNF...  
Edit Node... Double-Click  
Enter Group...  
Ungroup  
Sort Names...  
Enter Separator...

lock.v - Text Editor

module lock(  
input clk,  
output out  
output [6:  
  
wire re  
button I  
button I  
button I  
  
// state  
paramete  
paramete  
paramete  
paramete  
paramete  
paramete  
  
// imple  
reg [2:0  
always @  
begin  
if (re  
else @  
S\_RI  
S\_0: state <= b0 ? S\_0 : b1 ? S\_01 : state;  
S\_01: state <= b0 ? S\_01 : b1 ? S\_RESET : state;

Untitled1 - Waveform Editor

Ref: 0.0ns Time: 29.1ns Interval: 29.1ns

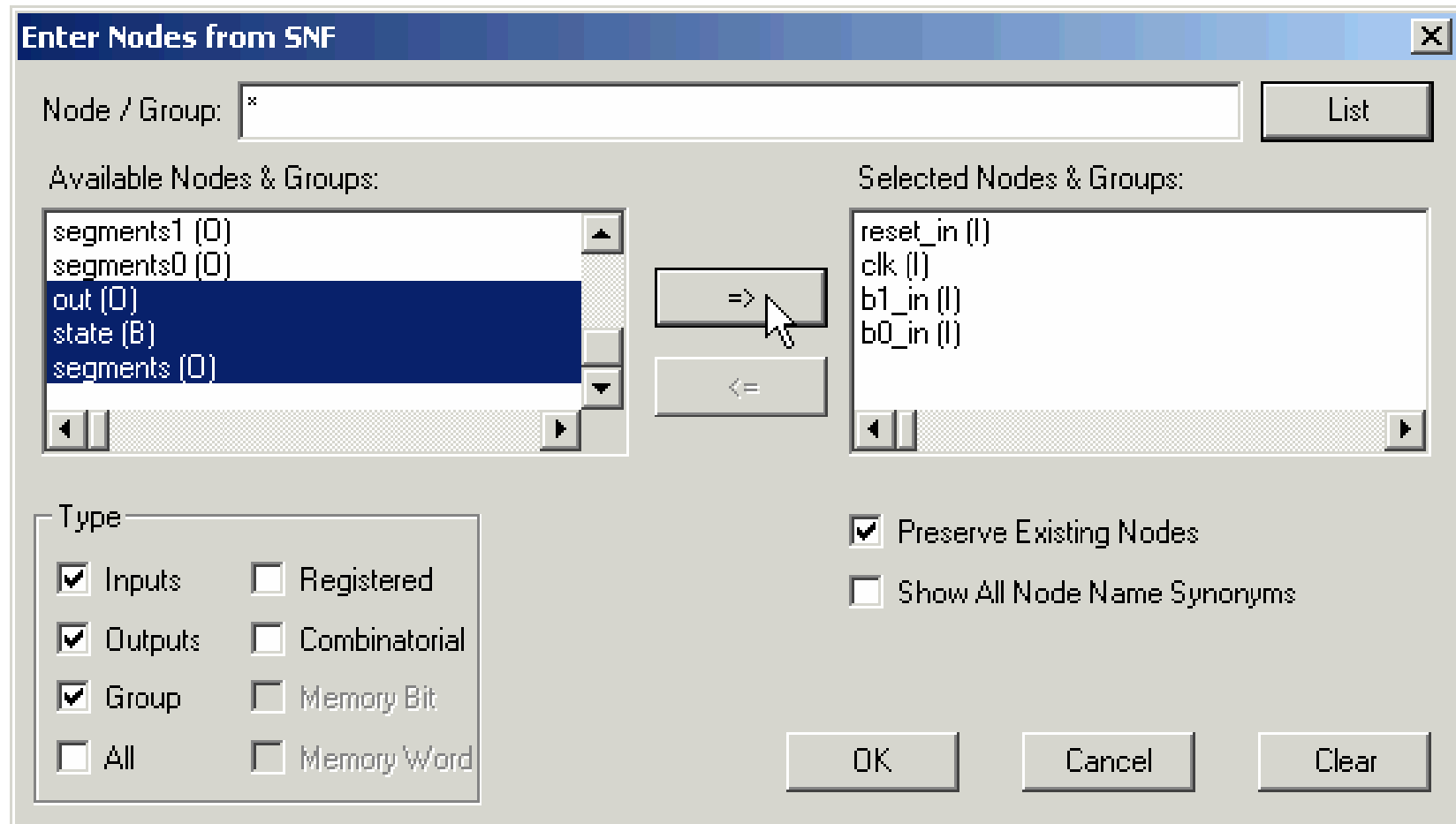
0.0ns

Name: Value: 100.0ns 200

Line 8 Col 12 INS

Adds nodes and groups from the project's SNF to the current file

# Step 3D: add I/O nodes



# Step 3E: set simulation end time

The screenshot shows the MAX+plus II software interface. The 'File' menu is open, and the 'End Time...' option is highlighted. The background shows a code editor with Verilog code and a timing diagram window.

MAX+plus II - c:\documents and settings\cjt\desktop\6111demo\lock

File Edit View Node Assign Utilities Options Window Help

Project

- New...
- Open... Ctrl+O
- Delete File...
- Retrieve...
- Close Ctrl+F4
- Save Ctrl+S
- Save As...
- Info... Ctrl+I
- End Time...**
- Compare...
- Import Vector File...
- Create Table File...
- Create Default Symbol
- Edit Symbol
- Create Default Include File
- Print... Ctrl+P
- Print Setup...
- Hierarchy
- MegaWizard Plug-In Manager
- Exit MAX+plus II Alt+F4

End: 1.0us Interval: 1.0us

100.0ns

```
always (
begin
  if (r
  else (
    S_RI
    S_0: state <= b0 ? S_0 : b1 ? S_01 : state;
```

Line 8 Col 12 INS

Specifies a new end time for the current file

# Step 3F: set input values, save

The screenshot displays the MAX+plus II software interface. The main window is titled 'lock.v - Text Editor' and contains the following Verilog code:

```
module lock(clk,reset_in,b0_in,b1_in,out,segments);
input clk,reset_in,b0_in,b1_in;
output out;
output [6:0] segments;

// state
parameter S_00 = 0;
parameter S_01 = 1;
parameter S_10 = 2;
parameter S_11 = 3;
parameter S_RESET = 4;

// impl
reg [2:0] state;
always @ (posedge clk)
begin
if (reset_in)
state <= S_RESET;
else
state <= b0_in ? S_00 : b1_in ? S_01 : state;
end

S_00: state <= b0_in ? S_00 : S_01;
S_01: state <= b0_in ? S_01 : S_00;
S_10: state <= b0_in ? S_10 : S_11;
S_11: state <= b0_in ? S_11 : S_10;
endmodule
```

The 'Untitled1 - Waveform Editor' window is overlaid on the code, showing a timing diagram from 7.1us to 10.0us. The diagram includes the following signals:

- clk**: A periodic clock signal.
- reset\_in**: A signal that transitions from 0 to 1 at approximately 7.2us.
- b1\_in**: A signal that transitions from 0 to 1 at approximately 7.5us.
- b0\_in**: A signal that transitions from 0 to 1 at approximately 7.8us.
- out**: A signal that is initially 'X' (unknown) and then transitions to 1 at approximately 8.2us.
- state**: A signal that is initially 'D X' (don't care) and then transitions to 'X' at approximately 8.2us.

The waveform editor also shows a table of signal values:

Name	Value
clk	1
reset_in	0
b1_in	0
b0_in	1
out	X
state	D X

The status bar at the bottom indicates 'Line 8 Col 12 INS' and a message: 'Overwrites selected waveform(s) with a high (1) logic level'.

# Step 3G: run simulation

The screenshot shows the MAX+plus II software interface. The 'Simulator' option is highlighted in the 'File' menu. The 'lock.scf - Waveform Editor' window is open, displaying a timing diagram for a simulation from 7.1us to 7.4us. The diagram shows signals for clk, reset\_in, b1\_in, b0\_in, out, and state. The clk signal is a periodic square wave. The reset\_in signal is a single pulse. The b1\_in and b0\_in signals are square waves. The out signal is a shaded area, and the state signal is a blue line with 'X' markers.

MAX+plus II - c:\documents and settings\cjt\desktop\6111 demo\lock

MAX+plus II File Edit View Node Assign Utilities Options Window Help

- Hierarchy Display
- Graphic Editor
- Symbol Editor
- Text Editor
- Waveform Editor
- Floorplan Editor
- Compiler
- Simulator**
- Timing Analyzer
- Programmer
- Message Processor

lock.scf - Waveform Editor

Start: 7.1us End: 7.4us Interval: 300.0ns

Name	Value
clk	1
reset_in	0
b1_in	0
b0_in	1
out	X
state	D X

```
// state
parameter
parameter
parameter
parameter
parameter
parameter

// impl
reg [2:0]
always
begin
  if (re
  else
    S_RI
    S_0: state <= b0 ? S_0 : b1 ? S_01 : state;
```

Line 8 Col 12 INS

Opens the Simulator window or brings it to the foreground

# Step 3H: examine simulation results

The screenshot displays the MAX+plus II software interface. The main window shows a timing simulation window titled "Simulator: Timing Simulation" with the following settings:

- Simulation Input: lock.scf
- Simulation Time: 10.0us
- Start Time: 0.0ns
- End Time: 10.0ns
- Use Device:
- Setup/Hold:
- Check Outputs:
- Oscillation: 0.0ns
- Glitch: 0.0ns

Below the settings is a progress bar and buttons for Start, Pause, Stop, and Open SCF. The background shows a code editor with the following code:

```
parameter S...
parameter S...
parameter S...
parameter S...

// implement
reg [2:0] st
always @ (p
begin
if (reset)
else case
S_RESET:
S_0: sta
```

The waveform viewer shows a timing diagram with the following signals and values:

Name	Value
clk	1
reset_in	0
b1_in	1
b0_in	0
out	0
state	D 4

The waveform viewer also shows a sequence of values for the state signal: 0, 1, 2, 3, 1, 2, 3, 4, 5. The time axis is labeled from 0 to 10.0us with major ticks every 2.0us.



# Step 4A: Assign→Device...

The screenshot shows the MAX+plus II software interface. The 'Device' dialog box is open, showing the 'Device Family' set to 'FLEX10K' and a list of devices with 'EPF10K10LC84-3' selected. Below the list, there are checkboxes for 'Show Only Fastest Speed Grades' (checked) and 'Maintain Current Synthesis Regardless of Device or Speed Grade Changes' (unchecked). The 'FLEX 10K / ACEX 1K Individual Device Options' dialog box is also open, showing various device options with checkboxes for 'User-Supplied Start-Up Clock (CLKUSR)', 'Auto-Restart Configuration on Frame Error', 'Release Clears Before Tri-States', 'Enable Chip-Wide Reset (DEV\_CLRn)', 'Enable Chip-Wide Output Enable (DEV\_DE)', 'Enable INIT\_DONE Output', 'Enable LOCK Output', 'MultiVolt I/O', 'Use Low-Voltage Configuration Device', and 'Use Configuration Device Pull-Up Resistor'. The 'Configuration Device' is set to 'EPC2LC20'. Below these options, there is a section for 'Dual-Purpose Configuration Pins' with a table of pin settings and a 'Configuration Scheme' dropdown set to 'Use Global Project Setting'. At the bottom, there are fields for 'FLEX / ACEX Device JTAG User Code' and 'Configuration Device JTAG User Code'. A code editor at the bottom shows Verilog code for a state transition diagram.

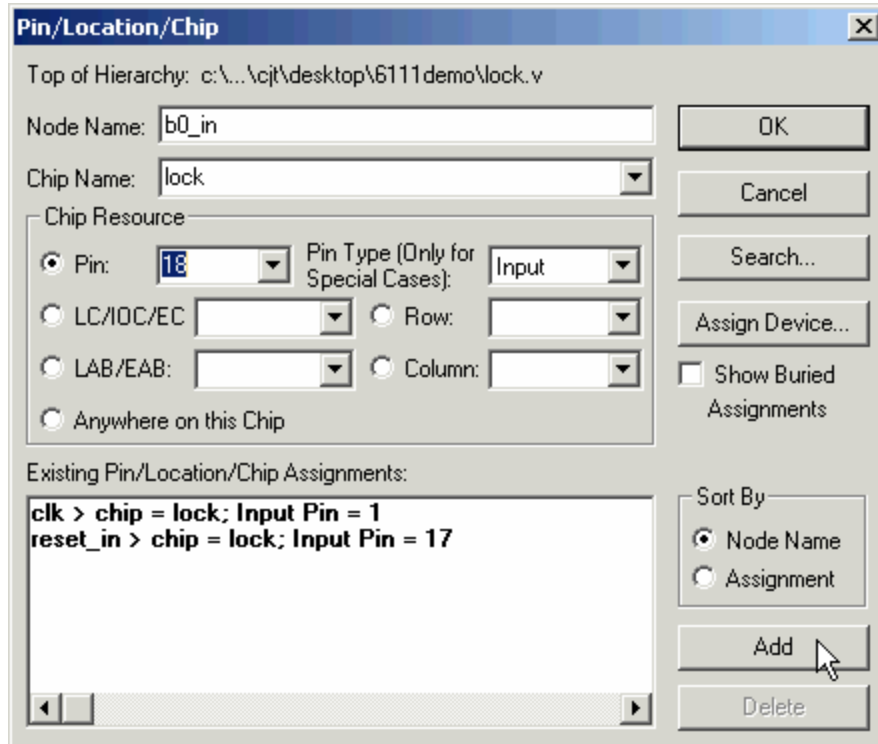
```

parameter S_0 = 1;
parameter S_01 = 2;
parameter S_010 = 3;
parameter S_0101 = 4;
parameter S_01011 = 5;

// implement state transition diagram
reg [2:0] state;
always @ (posedge clk)
begin
  if (reset) state <= S_RESET;
  else case (state)
    S_RESET: state <= b0 ? S_0 : b1 ? S_RESET : st
    S_0: state <= b0 ? S_0 : b1 ? S_01 : state;
    S_01: state <= b0 ? S_0 : b1 ? S_010 : S_RESET;
  endcase
end

```

# Step 4B: Assign→Pins...



Correlation between NuBus Pins and FLEX 10K Pins

NuBus Address	FLEX 10K10 Pin Number	FLEX 10K70 Pin Number
AD0	1 (CLK)	91 (CLK)
AD1	NONE	NONE
AD2	17	7
AD3	18	8
AD4	19	9
AD5	21	12
AD6	22	13
AD7	23	14
AD8	24	15
AD9	25	17
AD10	27	18
AD11	28	19
AD12	29	20
AD13	30	21

# Step 4C: recompile

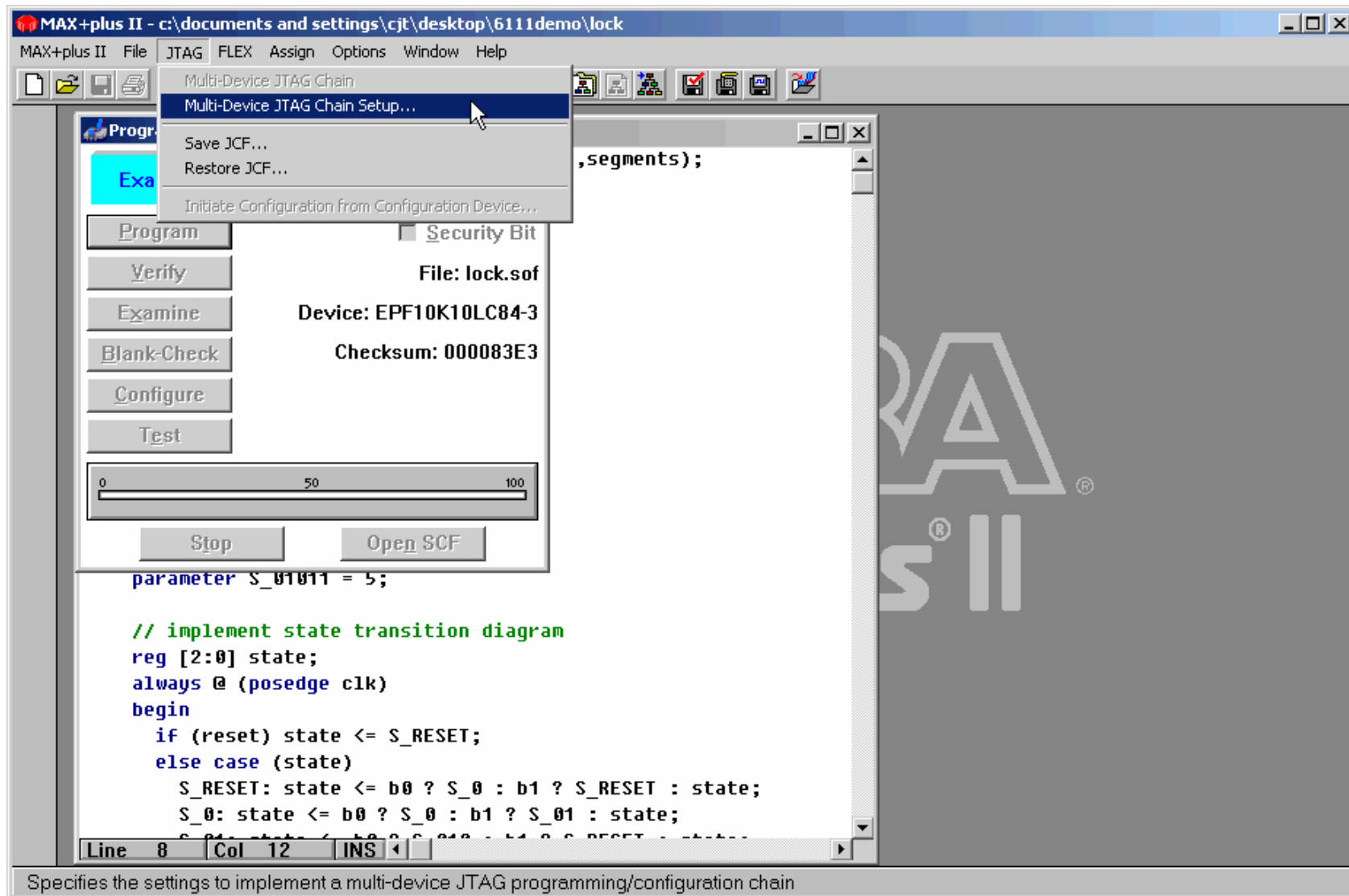
The screenshot displays the MAX+plus II software interface. The main window shows a text editor with the following Verilog code:

```
module lock(clk,reset_in,b0_in,b1_in,out,segments);
input clk,reset_in,b0_in,b1_in;
output out;
output [6:0] segments;

wire reset,b0,b1;
button b_reset(clk,reset_in,reset);
button
button
// sta
param
param
param
param
param
param
// imp
reg [2
always
begin
if (
else
S_
S_
S_
```

Overlaid on the text editor is the 'Compiler' window, which contains several tool buttons: Compiler Netlist Extractor, Database Builder, Logic Synthesizer, Partitioner, Fitter, Timing SNF Extractor, and Assembler. Below these buttons is a progress bar with a red line indicating the current progress, and 'Start' and 'Stop' buttons. The 'Messages - Compiler' window is also open, displaying the message: 'Info: Compiling project using Quartus Fitter technology.' At the bottom of the messages window, there are controls for 'Message' (0 of 1) and 'Locate' (0 of 0), along with a 'Locate All' button and a 'Help on Message' button. The status bar at the bottom of the text editor shows 'Line 8 Col 12 INS'.

# Step 4D: MAX+plusII→Programmer Set up JTAG chain



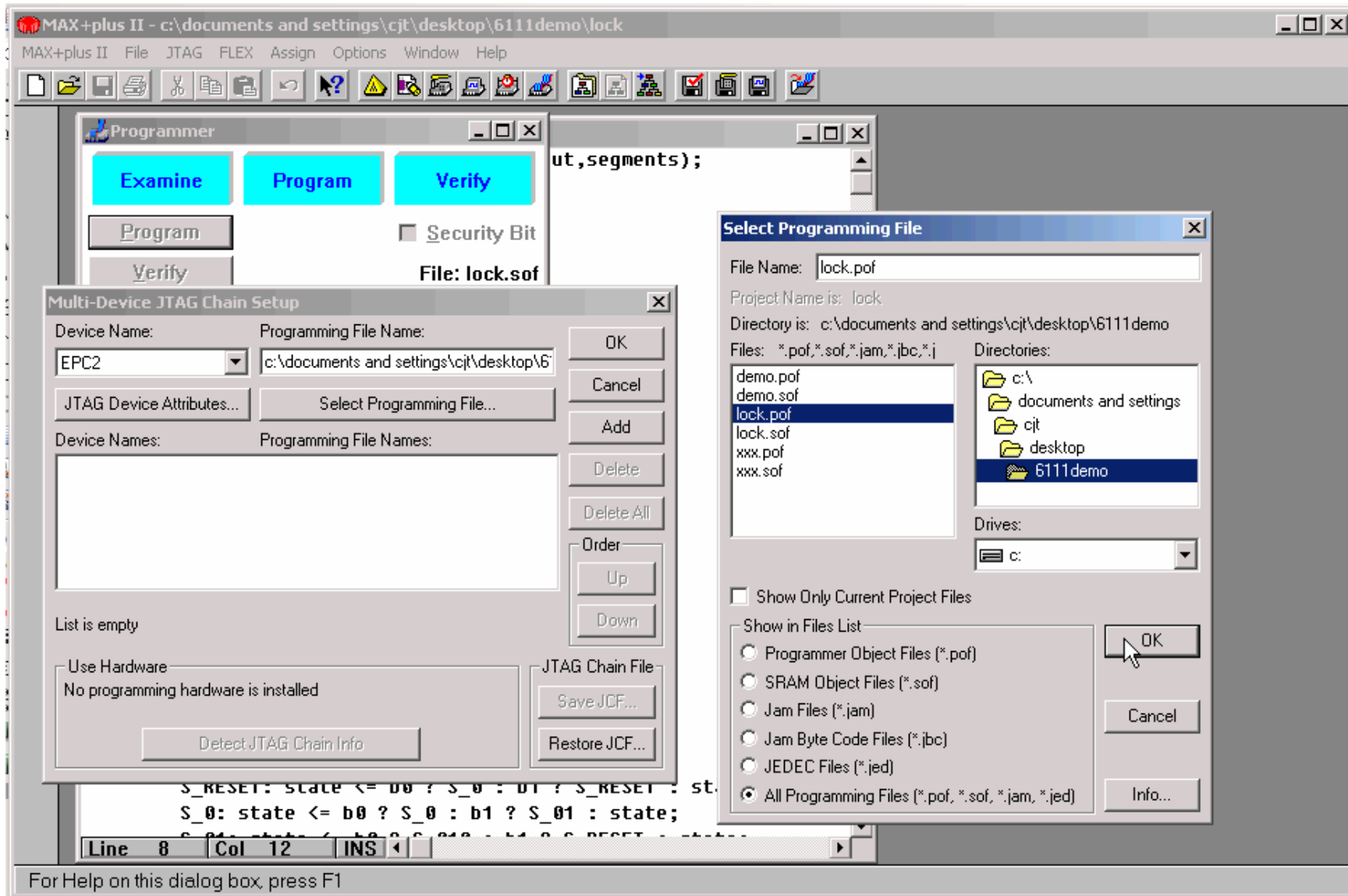
The screenshot shows the MAX+plus II software interface. The 'JTAG' menu is open, displaying options: 'Multi-Device JTAG Chain', 'Multi-Device JTAG Chain Setup...', 'Save JCF...', 'Restore JCF...', and 'Initiate Configuration from Configuration Device...'. The 'Multi-Device JTAG Chain Setup...' option is selected. Below the menu, the 'Program' button is highlighted. To the right, the configuration panel shows 'File: lock.sof', 'Device: EPF10K10LC84-3', and 'Checksum: 000083E3'. A progress bar is visible below the configuration panel. The main window displays Verilog code for a state transition diagram.

```
parameter S_01011 = 5;  
  
// implement state transition diagram  
reg [2:0] state;  
always @ (posedge clk)  
begin  
    if (reset) state <= S_RESET;  
    else case (state)  
        S_RESET: state <= b0 ? S_0 : b1 ? S_RESET : state;  
        S_0: state <= b0 ? S_0 : b1 ? S_01 : state;  
        S_01: state <= b0 ? S_01 : b1 ? S_RESET : state;  
    endcase  
end
```

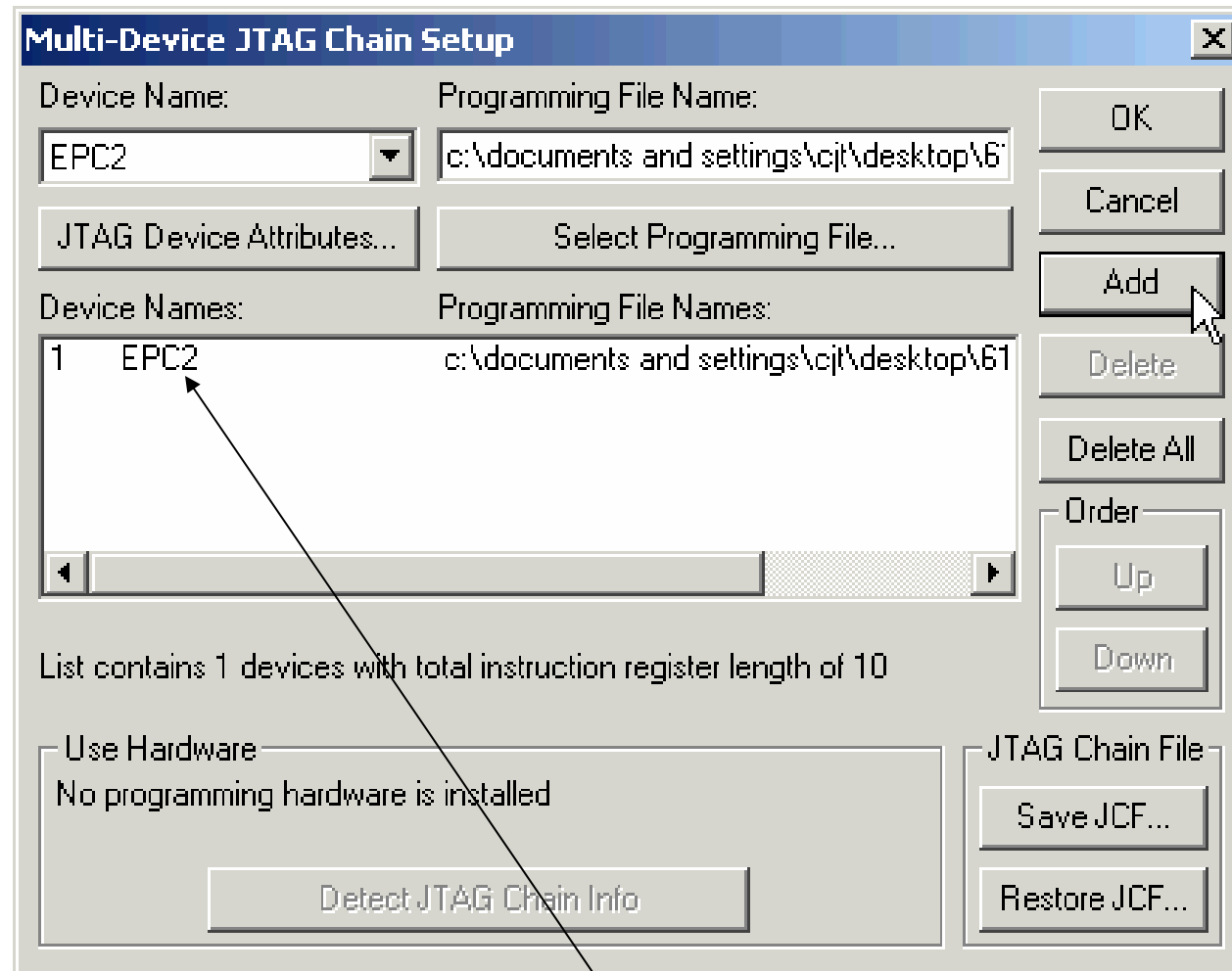
Line 8 Col 12 INS

Specifies the settings to implement a multi-device JTAG programming/configuration chain

# Step 4E: select programming file



# Step 4F: Add to list

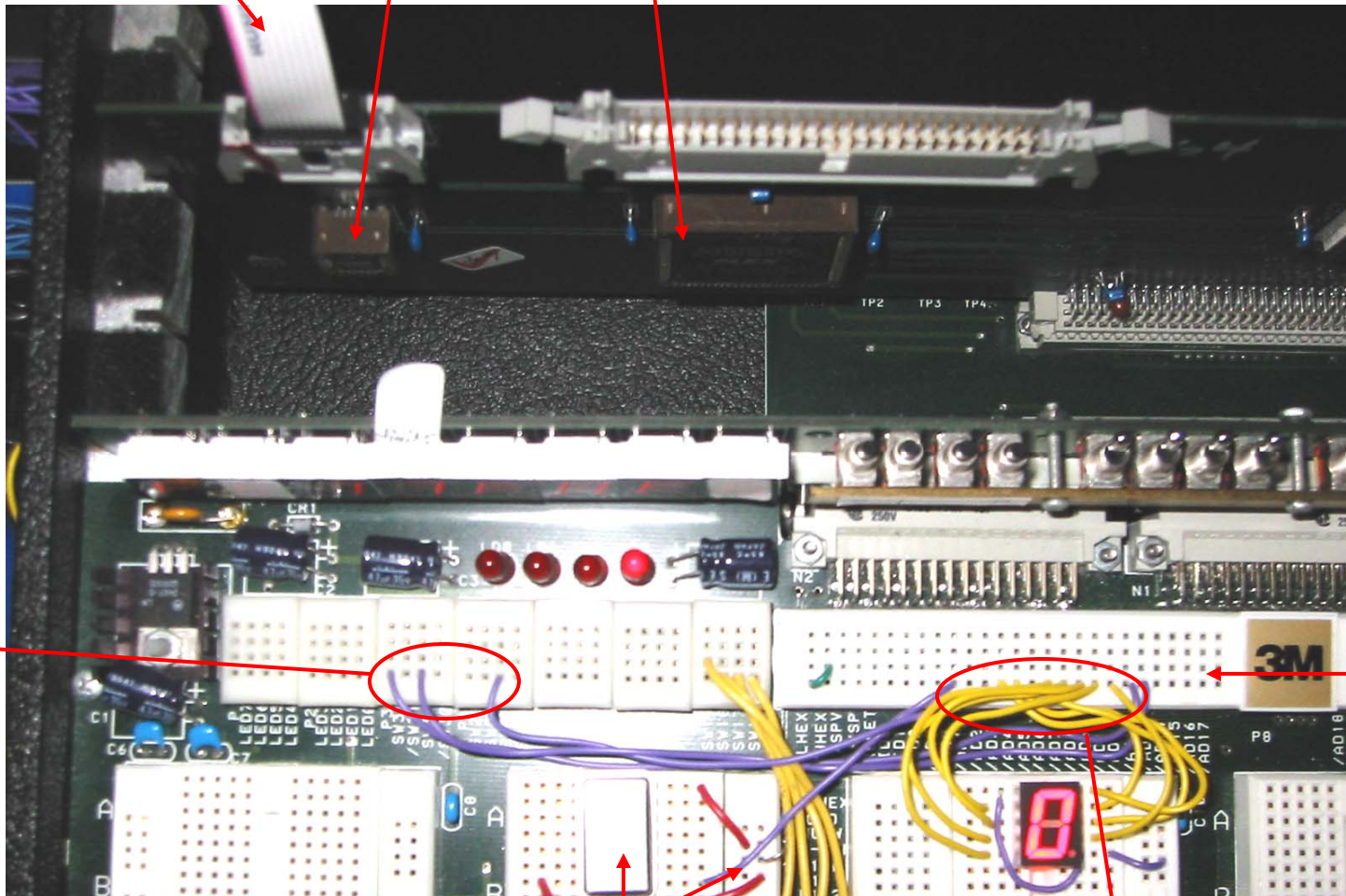


**We program the EPC2 serial flash memory whose contents is loaded into the Flex10K FPGA on power-up.**

# Step 4G: Attach ByteBlasterII cable

ByteBlasterII cable

EPC2LC20 serial flash RAM  
Flex10K10LC84 FPGA



Connections to buttons

Header with AD bus  
connections to FPGA pins

1.8432Mhz XTAL wired to /AD0 which  
connects to CLK input (pin 1) of FPGAs

FPGA I/Os wired to  
7-segment display

# Step 4H: program device, cycle power

