ECE 697J – Advanced Topics in Computer Networks

ACE Programming Model and SDK

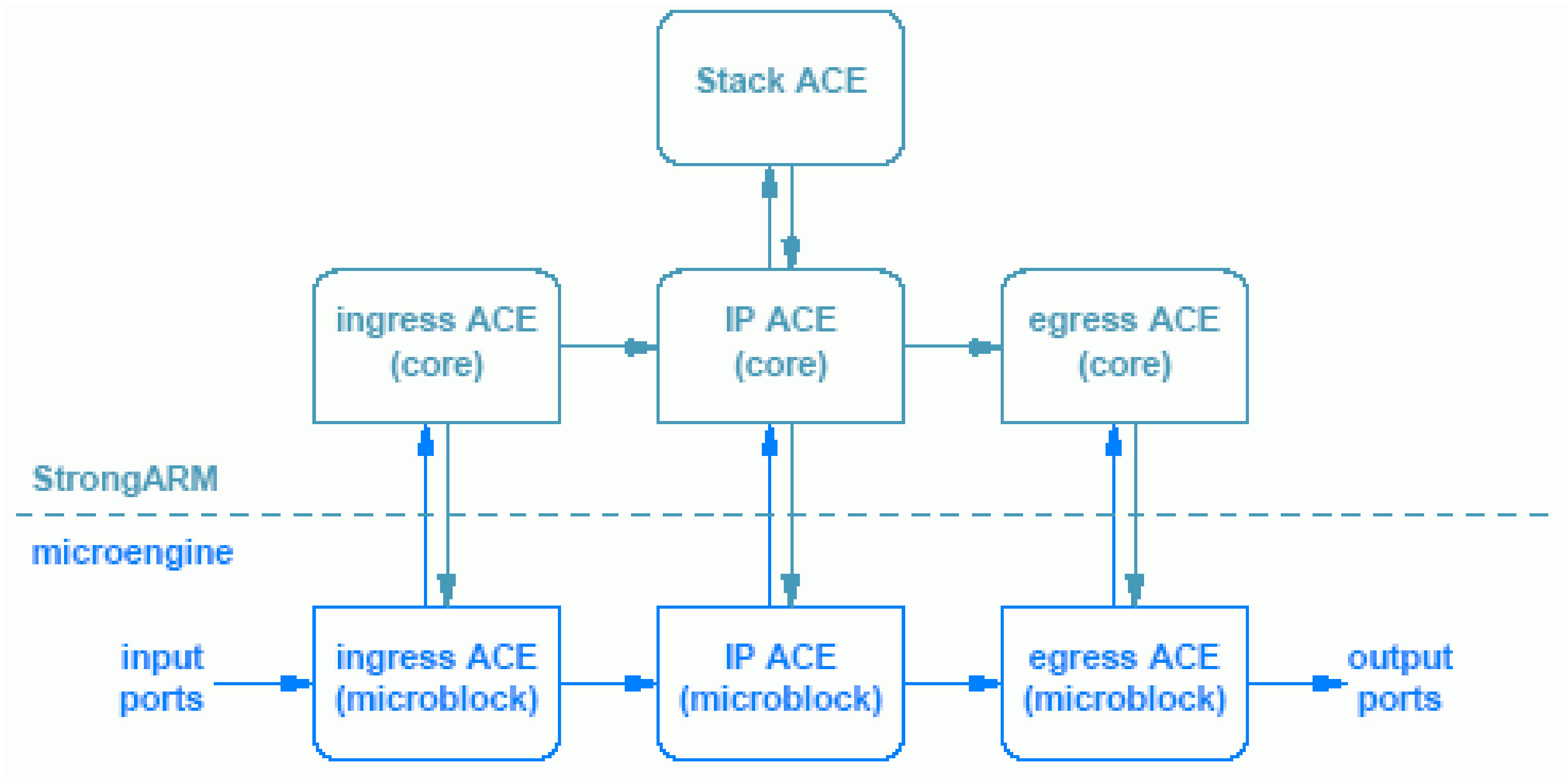
11/13/03

Overview

- Programming Model
  - Active Computing Element (ACE) Abstraction
  - Allocation of ACEs to microengines
  - Packet Queues
- Software Development Kit
  - Simulator
  - Example: IP forwarding
- Lab 2: IP forwarding and classification on IXP1200

Last Class

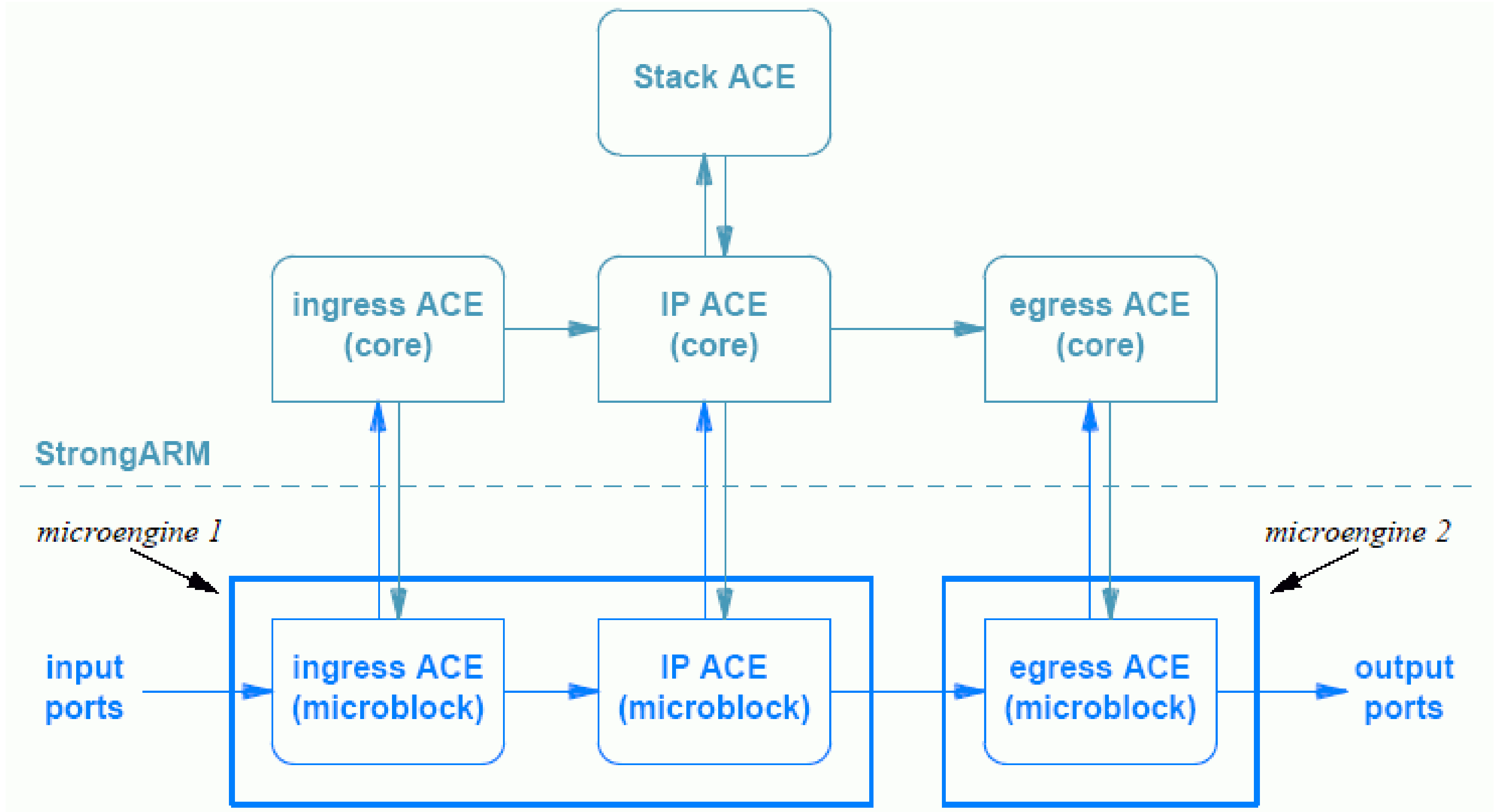
- Active Computing Element (ACE) abstraction:



Microengine Assignment

- Packet processing involves several microblocks
- How should microblocks be allocated to microengines?
  - One microblock per micorengine
  - Multiple microblocks per microengine (in pipeline)
  - Multiple pipelines on multiple microengines
- What are pros and cons?
  - Passing packets between microengines incurs overhead
  - Pipelining causes inefficiencies if blocks are not equal in size
  - Multiple blocks per microengine causes contention and requires more instruction storage
- Intel terminology: “microblock group”
  - Set of microblock running on one microengine

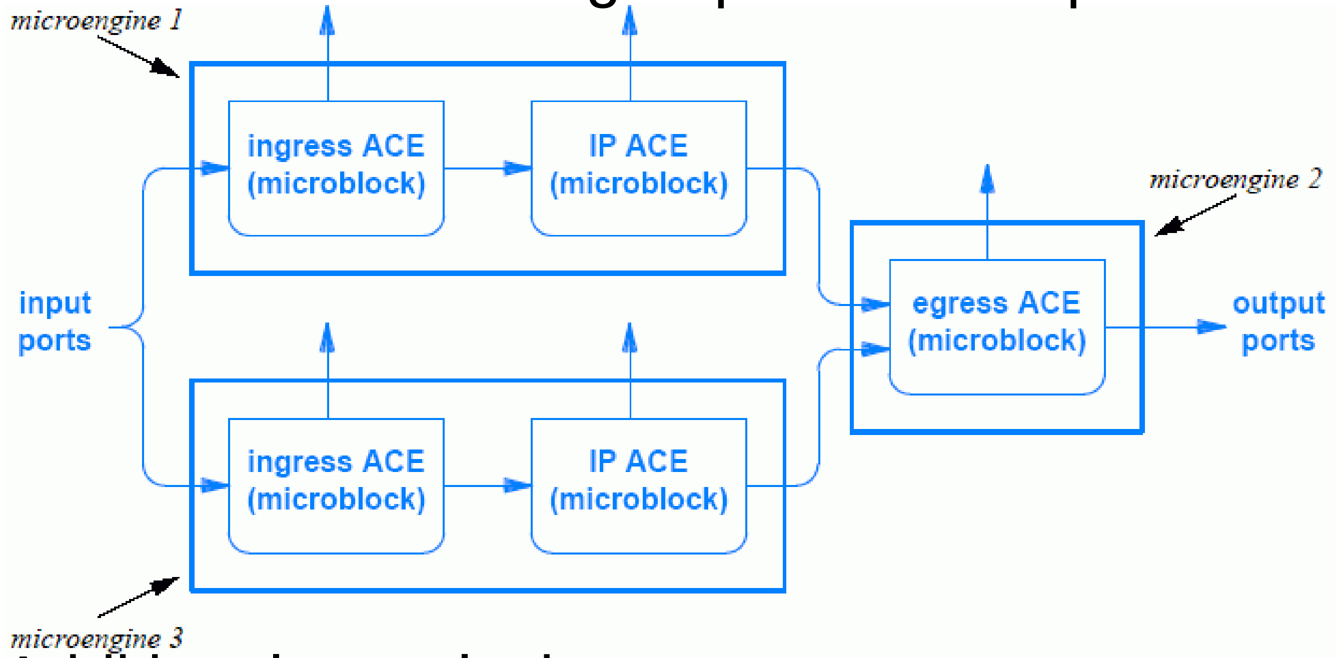
Microblock Groups



- Microblock groups can be replicated to increase parallelism

Microblock Group Replication

- Performance critical groups can be replicated:



- Additional complexity:
  - Single core component (not replicated) communicates with multiple groups
  - Multiple inputs, multiple output

Control of Packet Flow

- Packets require different processing blocks
  - IP requires different microblocks than ARP
  - Special packets get handed off to core
- "Dispatch Look" control packet flow among microblocks
  - Each thread runs its own dispatch loop
  - Infinite loop that grabs packets and hands them to microblocks
  - Return value from microblock determines the next step
- Invocation of microblock is similar to function call

Dispatch Loop

- Example:
  - Two microblocks (ingress + IP)

```
while (1) {
        Get next packet from input device(s);
        Invoke ingress microblock;
        if ( return code == 0 ) {
                Drop the packet;
        } else if ( return code == 1 ) {
                Send packet to ingress core component;
        } else { /* IP packet */
                Invoke IP microblock;
                if ( return code == 0 ) {
                        Drop packet;
                } else if ( return code == 1 ) {
                        Send packet to IP core component;
                } else {
                        Send packet to egress microblock;
                }
        }
}
```

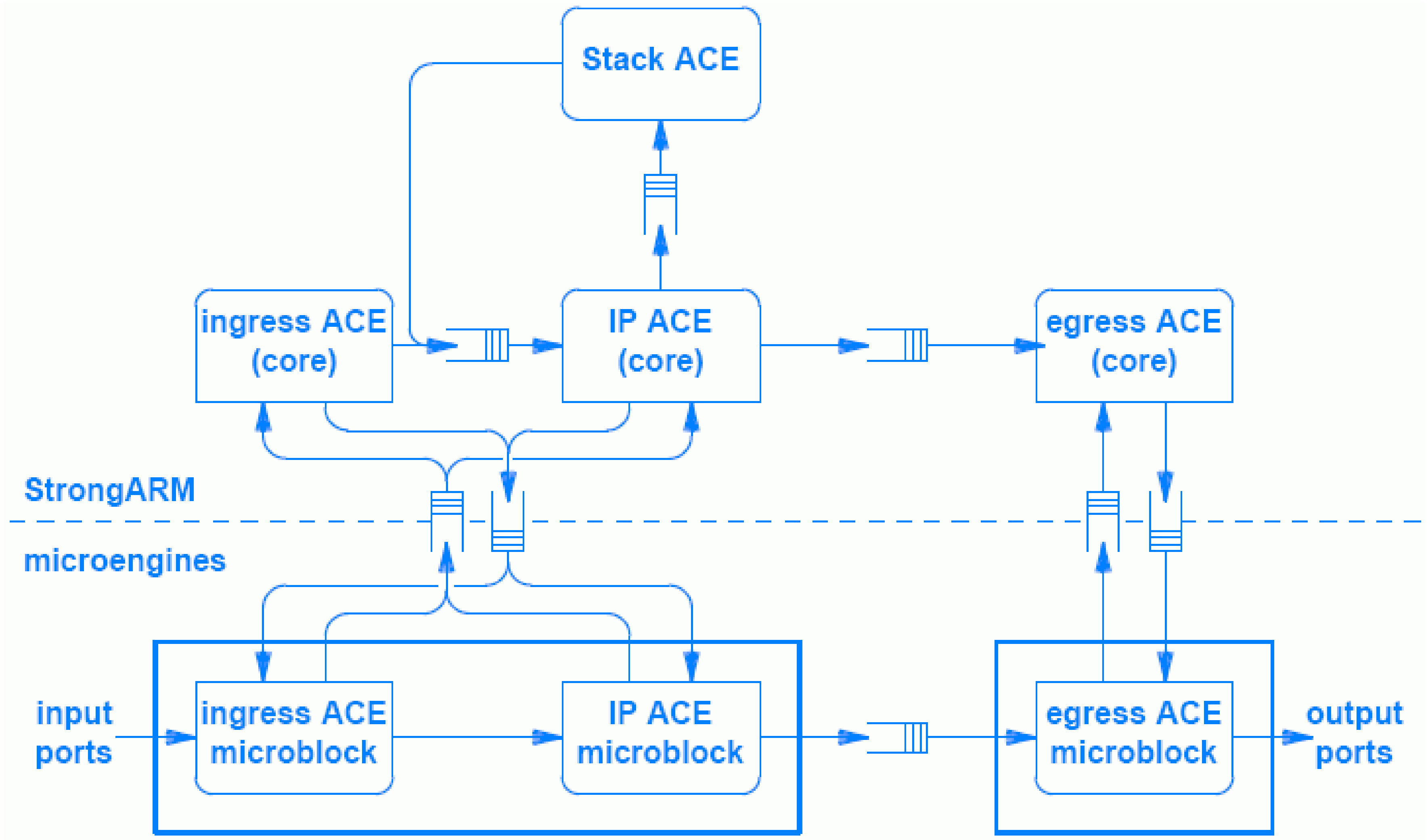
Dispatch Loop Conventions

- Parameters passed to microblock:
  - Buffer handle for frame that contains a packet
  - Set of state registers that contain information about the frame
  - A variable called dl_next_block in which return value gets stored
- State registers:
  - Information about packet: length
  - Information generated by software: classification result
  - Registers can be changed by microblock
- Return values:
  - Meaning assigned by programmer
  - Conventions: zero = “drop packet”, other values for “pass on” and “send to core” etc.

Packet Queues

- Packet flow depends on packet data
- Processing time depends on packet data
- Packet movement can’t be predicted
  - Microblocks need to continue processing without waiting
- Packets need to be buffered
  - “Communication Queues”
  - Unidirectional FIFO (yes, really FIFO)
  - Bidirectional communication requires two queues
- Also between microblocks and core
  - Single queue for all microblock group instances
  - Uses exception mechanism “IX_EXCEPTION”
  - Exception handler in core determines further steps

Packet Queue Example

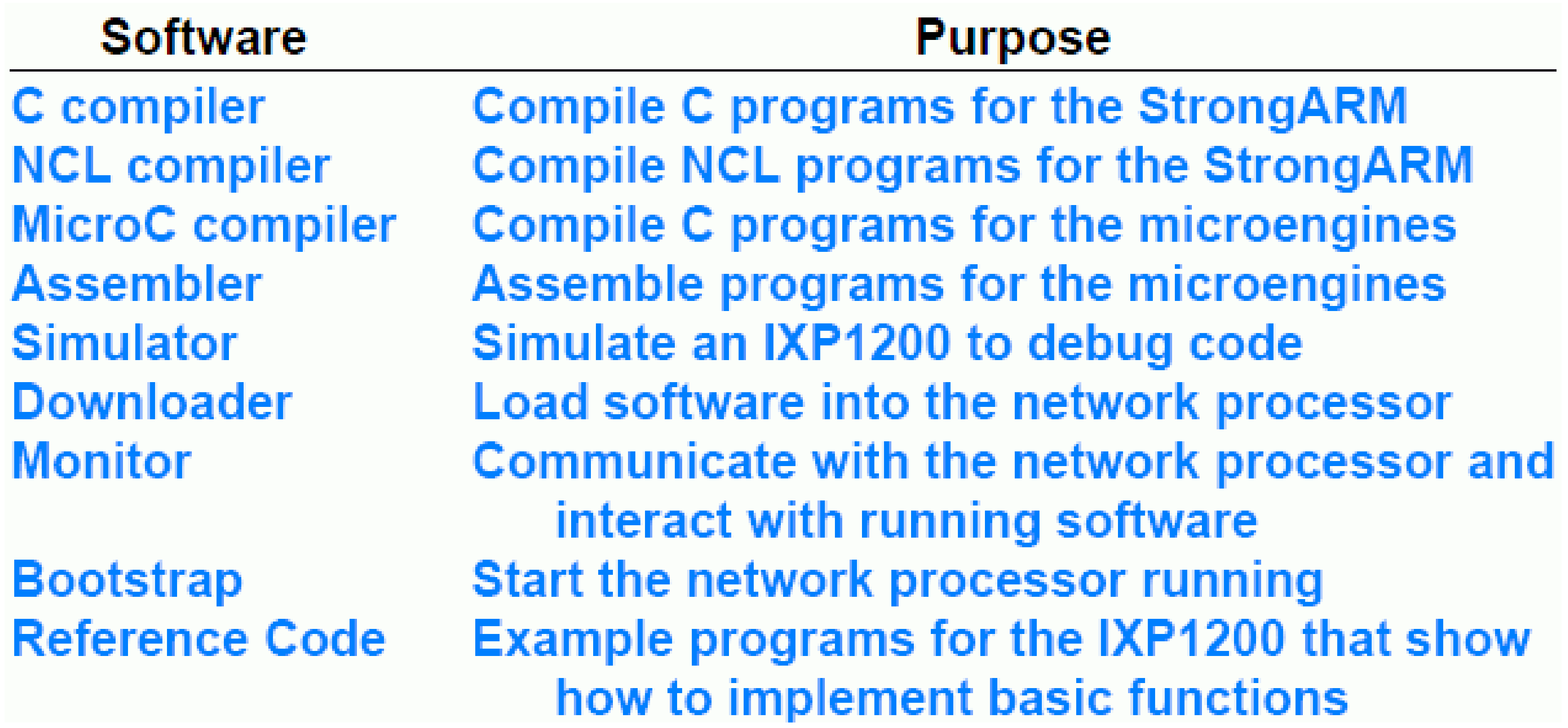


Crosscalls

- Mechanism for non-packet communication between ACEs
  - Similar to remote procedure calls and remote method invocations
- Caller and callee need to agree on parameters
  - Interface Definition Language (IDL) specifies details
  - IDL compiler creates “stubs” to handle marshaling
- Types of crosscalls
  - Deferred: caller does not block, asynchronous notification
  - Oneway: caller does not block, no return value
  - Twoway: caller blocks, callee returns value
- ACEs are prohibited from twoway calls
  - No blocking allowed
- Other control software (non-ACE) may use all types

SDK

- Software Development Kit:



Software	Purpose
C compiler	Compile C programs for the StrongARM
NCL compiler	Compile NCL programs for the StrongARM
MicroC compiler	Compile C programs for the microengines
Assembler	Assemble programs for the microengines
Simulator	Simulate an IXP1200 to debug code
Downloader	Load software into the network processor
Monitor	Communicate with the network processor and interact with running software
Bootstrap	Start the network processor running
Reference Code	Example programs for the IXP1200 that show how to implement basic functions

Software Setup

Figure 3: Linux software configuration for the Intel IXDP1200 Advanced Development Platform using only a Windows NT development environment

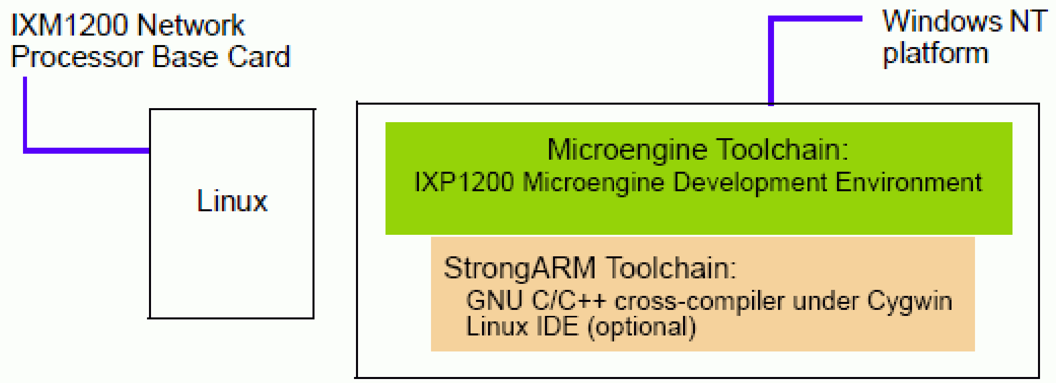
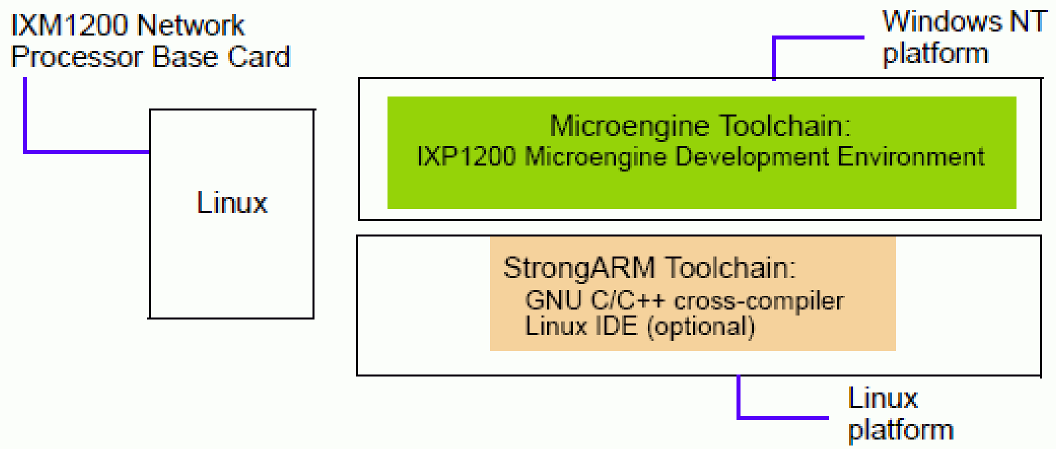

IXM1200 Network Processor Base Card
Linux
Windows NT platform
Microengine Toolchain:
IXP1200 Microengine Development Environment
StrongARM Toolchain:
GNU C/C++ cross-compiler under Cygwin
Linux IDE (optional)

Figure 4: Linux software configuration for the Intel IXDP1200 Advanced Development Platform using both Windows NT and Linux development environments

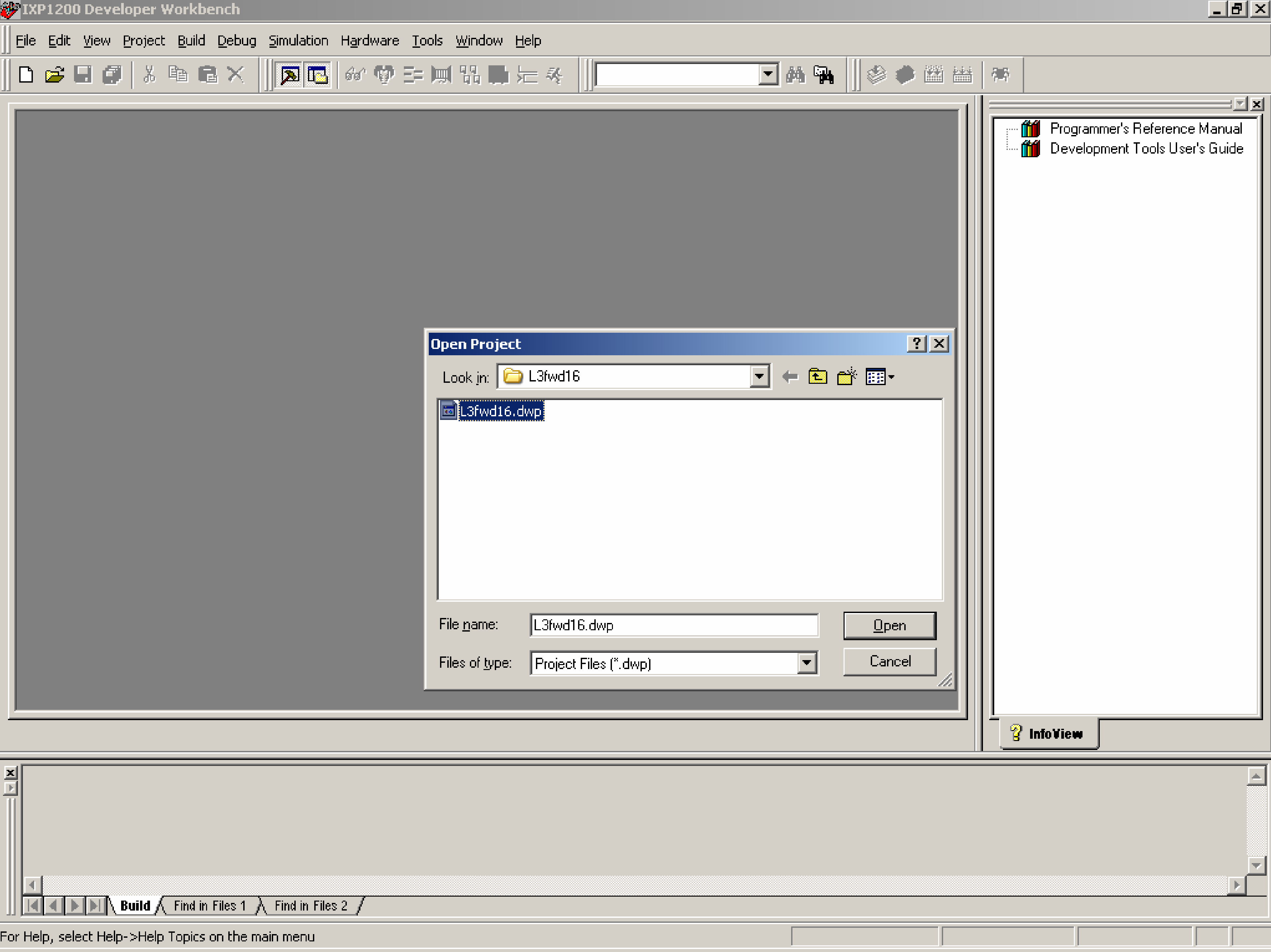


Simulator

- Cycle-accurate simulation of IXP1200
- Allows for easy experimentation
  - Packet generator
  - Visualization for thread behavior, memory accesses
  - Runs under Windows
- We will use simulator for Lab 2
  - Part I: run existing IP forwarding example, collect statistics
  - Part II: make a minor modification for classification
- We have lab machines set up for you
  - You can also install simulator on your own machine (big!)

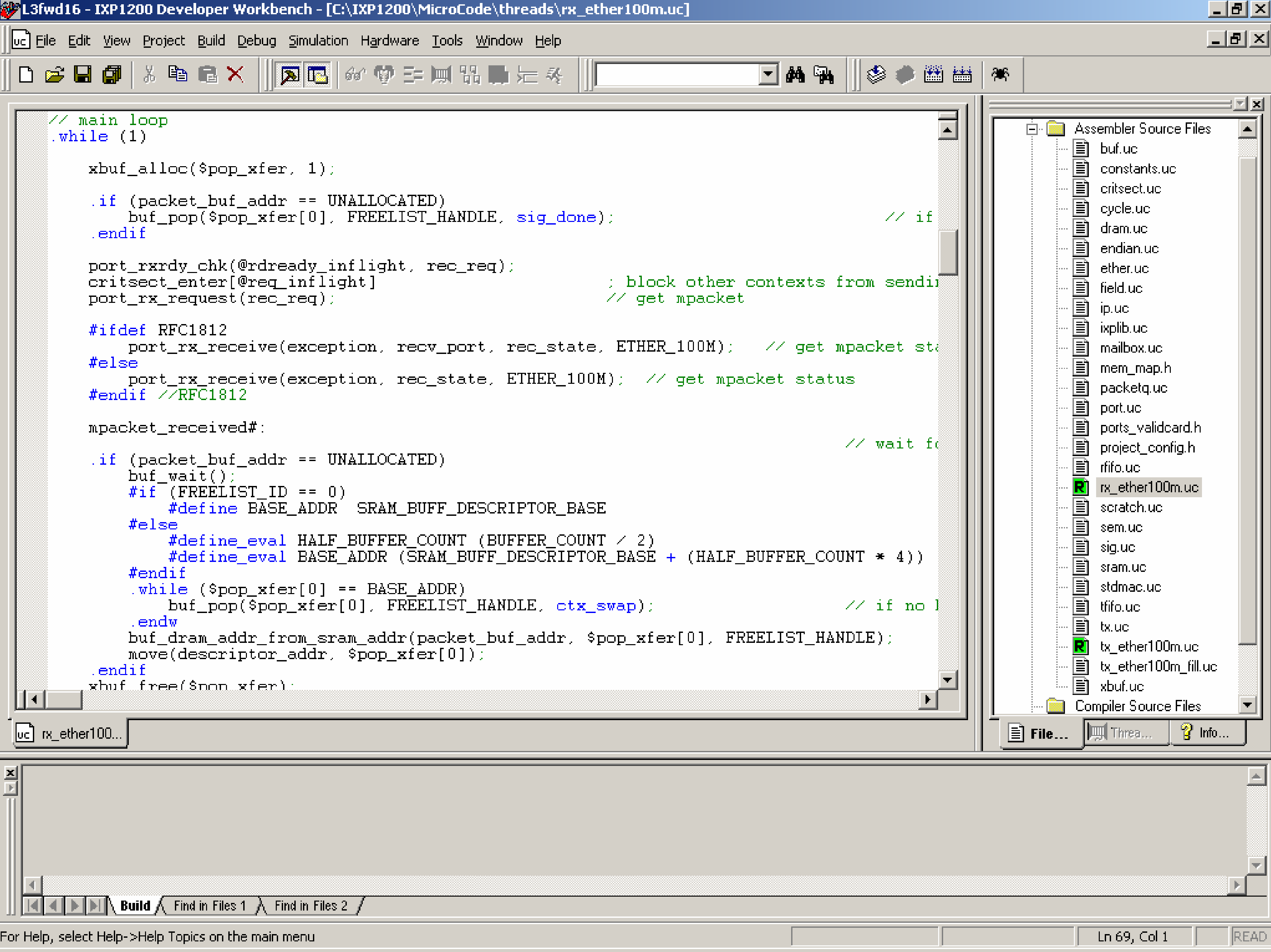
IP Forwarding Example

- Full-blown RFC1812-compliant IP forwarding
  - Lots of special cases
  - Look for main program structure
  - 4 uE for IP processing (0-3)
  - 3 uE for output queuing (4-5)
- Run program and collect workload statistics
  - Thread behavior
  - Memory accesses
  - Instruction coverage
  - Etc.


IXP1200 Developer Workbench
File Edit View Project Build Debug Simulation Hardware Tools Window Help
Programmer's Reference Manual
Development Tools User's Guide
InfoView
Open Project
Look in: L3fwd16
L3fwd16.dwp
Open
Cancel
Build
Find in Files 1
Find in Files 2

File name: L3fwd16.dwp
Files of type: Project Files (*.dwp)

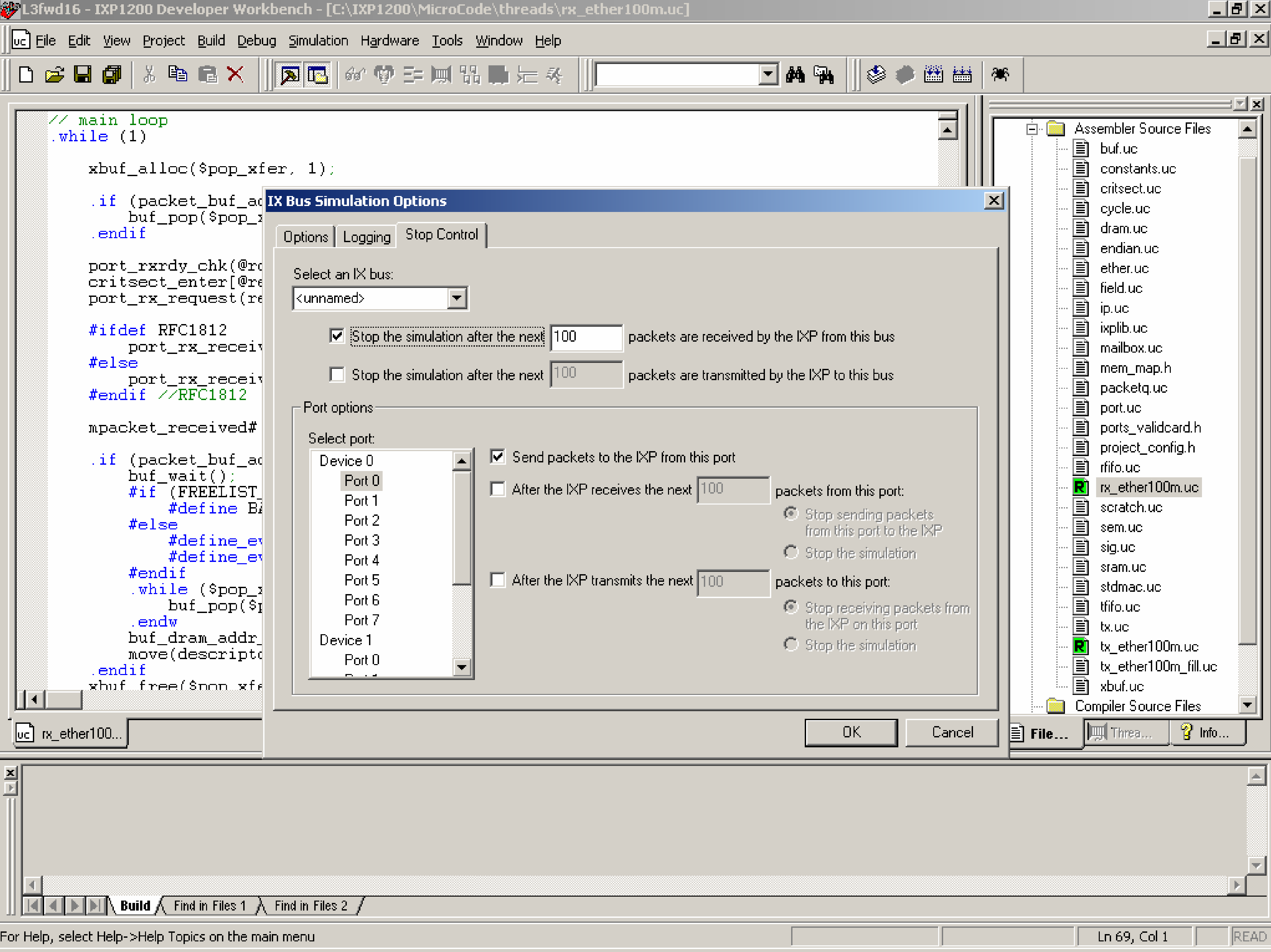
For Help, select Help->Help Topics on the main menu



```
// main loop
.while (1)

    xbuf_alloc($pop_xfer, 1);

    .if (packet_buf_addr == UNALLOCATED)
        buf_pop($pop_xfer[0], FREELIST_HANDLE, sig_done);                          // if
    .endif

    port_rxrdy_chk(@rdready_inflight, rec_req);
    critsect_enter[@req_inflight]                       ; block other contexts from sendi
    port_rx_request(rec_req);                           // get mpacket

    #ifdef RFC1812
        port_rx_receive(exception, recv_port, rec_state, ETHER_100M);   // get mpacket sta
    #else
        port_rx_receive(exception, rec_state, ETHER_100M);  // get mpacket status
    #endif //RFC1812

    mpacket_received#:
                                                                                // wait f
    .if (packet_buf_addr == UNALLOCATED)
        buf_wait();
        #if (FREELIST_ID == 0)
            #define BASE_ADDR  SRAM_BUFF_DESCRIPTOR_BASE
        #else
            #define_eval HALF_BUFFER_COUNT (BUFFER_COUNT / 2)
            #define_eval BASE_ADDR (SRAM_BUFF_DESCRIPTOR_BASE + (HALF_BUFFER_COUNT * 4))
        #endif
        .while ($pop_xfer[0] == BASE_ADDR)
            buf_pop($pop_xfer[0], FREELIST_HANDLE, ctx_swap);                   // if no
        .endw
        buf_dram_addr_from_sram_addr(packet_buf_addr, $pop_xfer[0], FREELIST_HANDLE);
        move(descriptor_addr, $pop_xfer[0]);
    .endif
    xbuf_free($pop_xfer);
```

For Help, select Help->Help Topics on the main menu

Ln 69, Col 1 READ



L3fwd16 - IXP1200 Developer Workbench - [C:\IXP1200\MicroCode\threads\rx_ether100m.uc]

```
// main loop
.while (1)

    xbuf_alloc($pop_xfer, 1);

    .if (packet_buf_ac
        buf_pop($pop_x
    .endif

    port_rxrdy_chk(@rc
    critsect_enter[@re
    port_rx_request(re

    #ifdef RFC1812
        port_rx_receiv
    #else
        port_rx_receiv
    #endif //RFC1812

    mpacket_received#

    .if (packet_buf_ac
        buf_wait();
        #if (FREELIST_
            #define BA
        #else
            #define_ev
            #define_ev
        #endif
        .while ($pop_x
            buf_pop($p
        .endw
        buf_dram_addr_
        move(descripto
    .endif
    xbuf_free($pop_xfe
```

IX Bus Simulation Options

Options | Logging | Stop Control

Select an IX bus: <unnamed>

[✓] Stop the simulation after the next 100 packets are received by the IXP from this bus

[ ] Stop the simulation after the next 100 packets are transmitted by the IXP to this bus

Port options

Select port: Device 0 — Port 0, Port 1, Port 2, Port 3, Port 4, Port 5, Port 6, Port 7; Device 1 — Port 0

[✓] Send packets to the IXP from this port

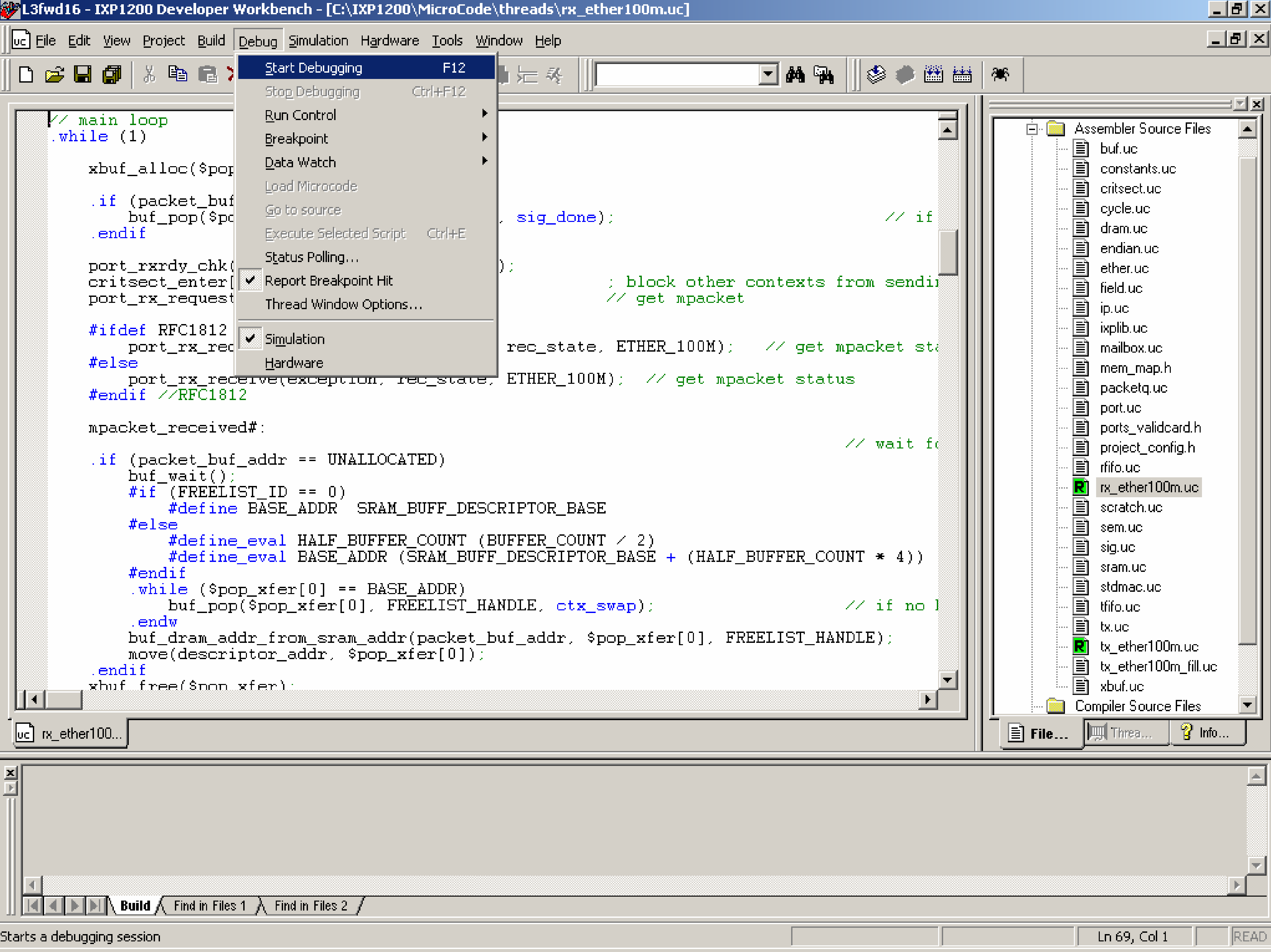
[ ] After the IXP receives the next 100 packets from this port:
- (●) Stop sending packets from this port to the IXP
- ( ) Stop the simulation

[ ] After the IXP transmits the next 100 packets to this port:
- (●) Stop receiving packets from the IXP on this port
- ( ) Stop the simulation

OK | Cancel

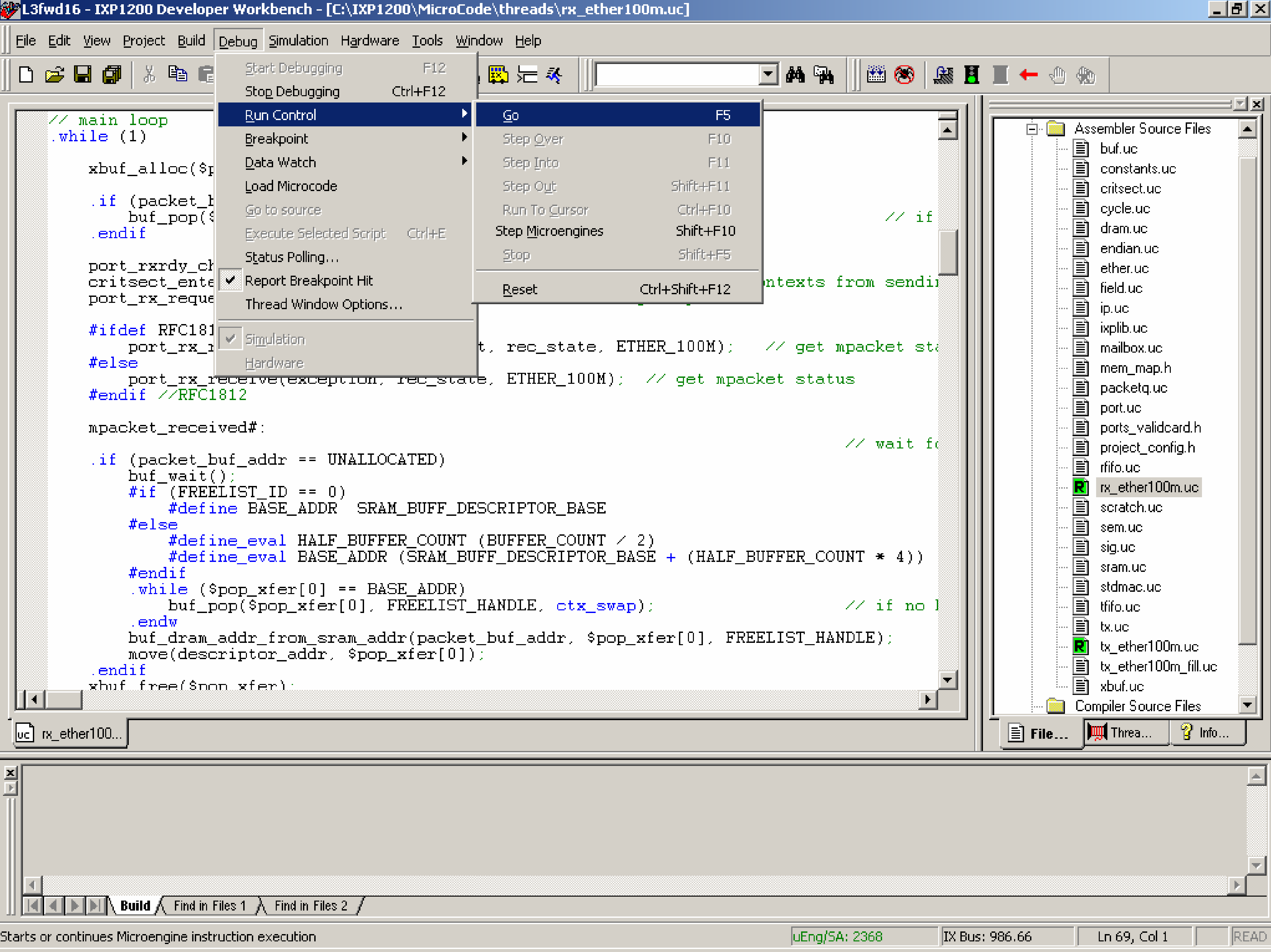
Assembler Source Files: buf.uc, constants.uc, critsect.uc, cycle.uc, dram.uc, endian.uc, ether.uc, field.uc, ip.uc, ixplib.uc, mailbox.uc, mem_map.h, packetq.uc, port.uc, ports_validcard.h, project_config.h, rfifo.uc, rx_ether100m.uc, scratch.uc, sem.uc, sig.uc, sram.uc, stdmac.uc, tfifo.uc, tx.uc, tx_ether100m.uc, tx_ether100m_fill.uc, xbuf.uc
Compiler Source Files

For Help, select Help->Help Topics on the main menu | Ln 69, Col 1 | READ



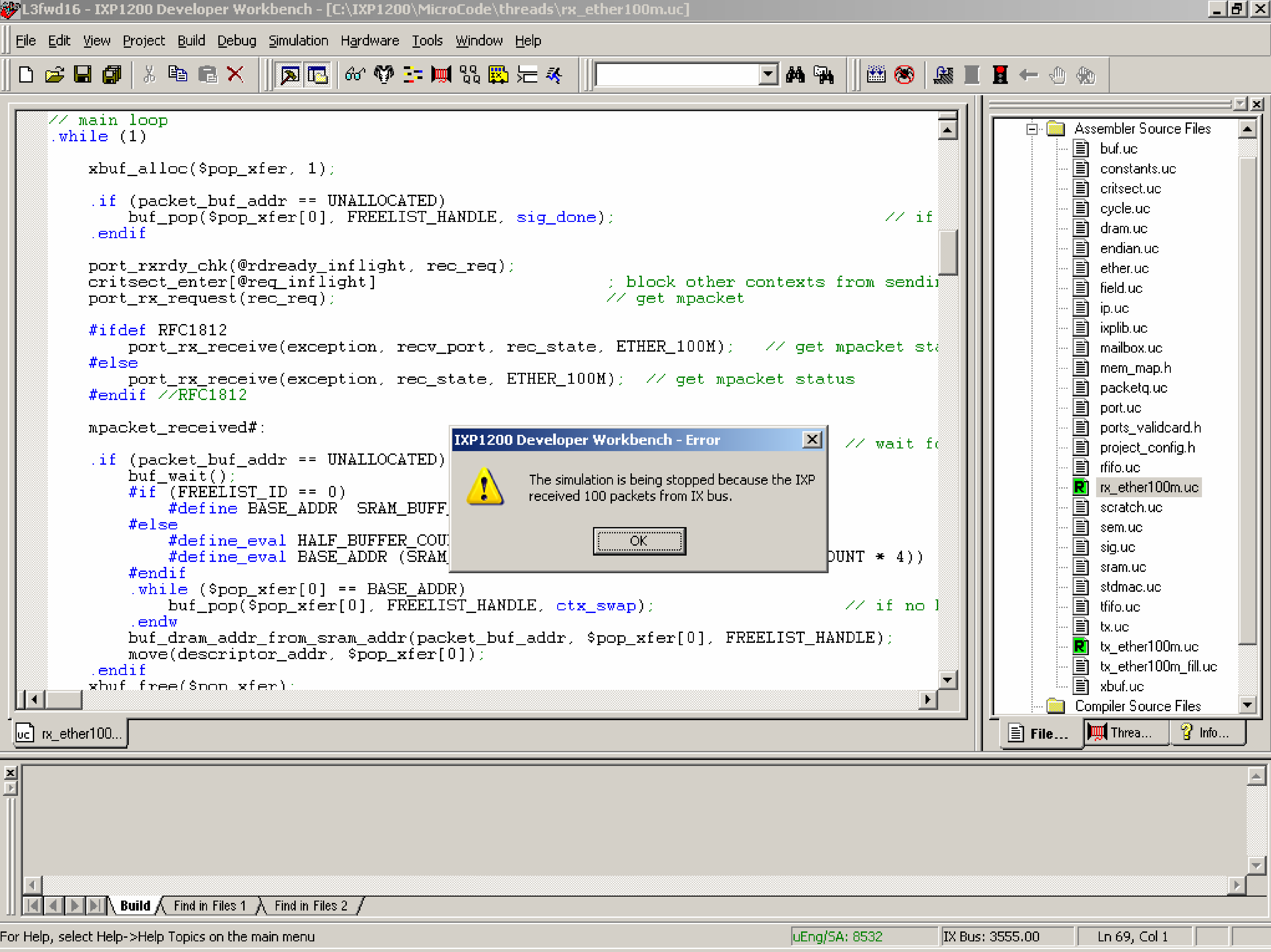
```
// main loop
.while (1)

    xbuf_alloc($pop

    .if (packet_buf
        buf_pop($pc                          , sig_done);                          // if
    .endif

    port_rxrdy_chk(                          );
    critsect_enter(                                  ; block other contexts from sendi
    port_rx_request                                  // get mpacket

    #ifdef RFC1812
        port_rx_rec                          rec_state, ETHER_100M);   // get mpacket sta
    #else
        port_rx_receive(exception, rec_state, ETHER_100M);  // get mpacket status
    #endif //RFC1812

    mpacket_received#:
                                                                                   // wait fo
    .if (packet_buf_addr == UNALLOCATED)
        buf_wait();
        #if (FREELIST_ID == 0)
            #define BASE_ADDR  SRAM_BUFF_DESCRIPTOR_BASE
        #else
            #define_eval HALF_BUFFER_COUNT (BUFFER_COUNT / 2)
            #define_eval BASE_ADDR (SRAM_BUFF_DESCRIPTOR_BASE + (HALF_BUFFER_COUNT * 4))
        #endif
        .while ($pop_xfer[0] == BASE_ADDR)
            buf_pop($pop_xfer[0], FREELIST_HANDLE, ctx_swap);                     // if no l
        .endw
        buf_dram_addr_from_sram_addr(packet_buf_addr, $pop_xfer[0], FREELIST_HANDLE);
        move(descriptor_addr, $pop_xfer[0]);
    .endif
    xbuf_free($pop_xfer);
```

Starts a debugging session



```
// main loop
.while (1)

    xbuf_alloc($p

    .if (packet_b
        buf_pop($                                                            // if
    .endif

    port_rxrdy_ch
    critsect_ente                              ntexts from sendi
    port_rx_reque

    #ifdef RFC181
        port_rx_r                    t, rec_state, ETHER_100M);    // get mpacket sta
    #else
        port_rx_receive(exception, rec_state, ETHER_100M);  // get mpacket status
    #endif //RFC1812

    mpacket_received#:
                                                                      // wait fo
    .if (packet_buf_addr == UNALLOCATED)
        buf_wait();
        #if (FREELIST_ID == 0)
            #define BASE_ADDR  SRAM_BUFF_DESCRIPTOR_BASE
        #else
            #define_eval HALF_BUFFER_COUNT (BUFFER_COUNT / 2)
            #define_eval BASE_ADDR (SRAM_BUFF_DESCRIPTOR_BASE + (HALF_BUFFER_COUNT * 4))
        #endif
        .while ($pop_xfer[0] == BASE_ADDR)
            buf_pop($pop_xfer[0], FREELIST_HANDLE, ctx_swap);          // if no l
        .endw
        buf_dram_addr_from_sram_addr(packet_buf_addr, $pop_xfer[0], FREELIST_HANDLE);
        move(descriptor_addr, $pop_xfer[0]);
    .endif
    xbuf_free($pop_xfer);
```

Starts or continues Microengine instruction execution


L3fwd16 - IXP1200 Developer Workbench - [C:\IXP1200\MicroCode\threads\rx_ether100m.uc]

File Edit View Project Build Debug Simulation Hardware Tools Window Help

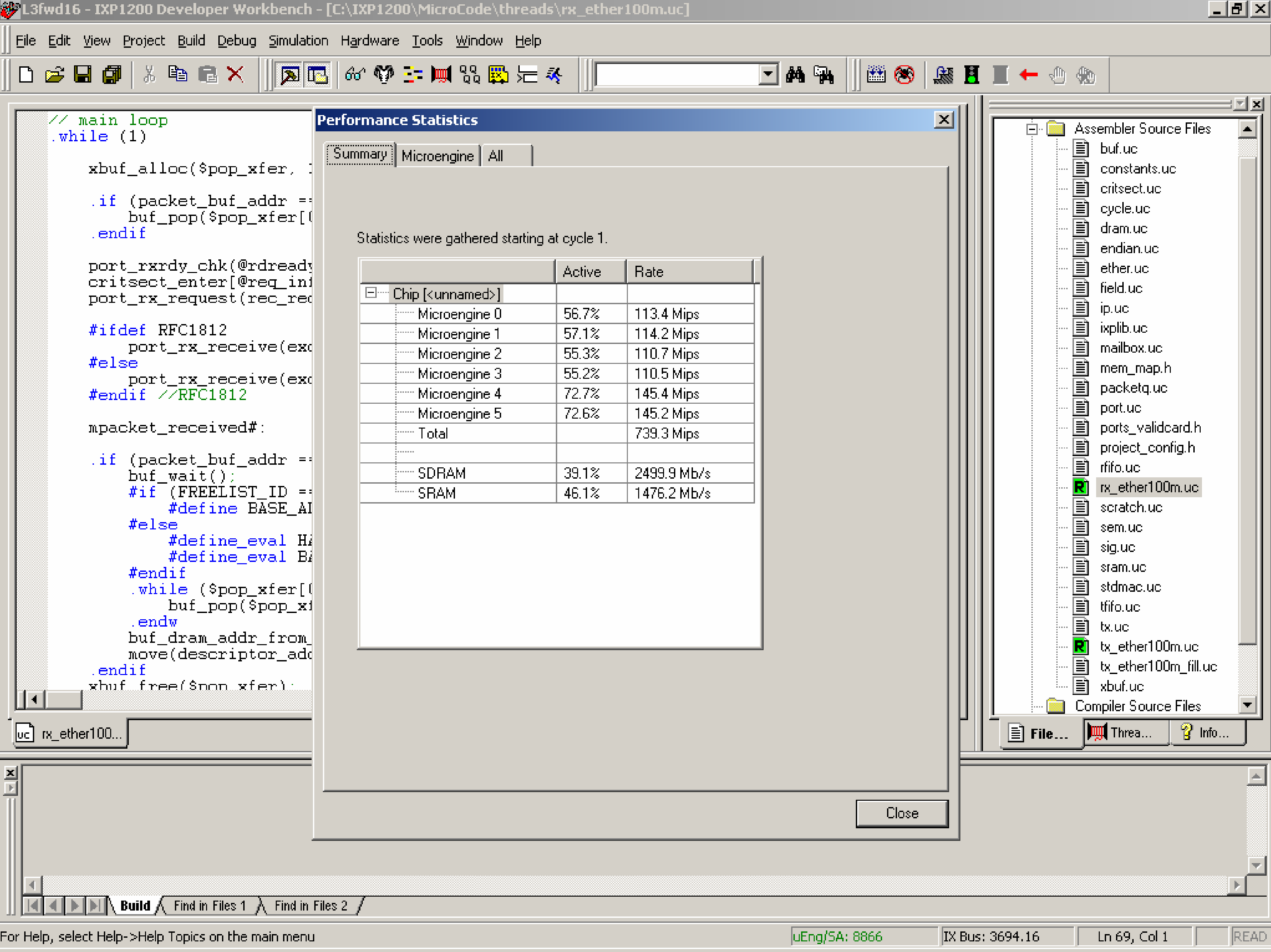
```
// main loop
.while (1)

    xbuf_alloc($pop_xfer, 1);

    .if (packet_buf_addr == UNALLOCATED)
        buf_pop($pop_xfer[0], FREELIST_HANDLE, sig_done);                    // if
    .endif

    port_rxrdy_chk(@rdready_inflight, rec_req);
    critsect_enter[@req_inflight]                          ; block other contexts from sendi
    port_rx_request(rec_req);                              // get mpacket

    #ifdef RFC1812
        port_rx_receive(exception, recv_port, rec_state, ETHER_100M);   // get mpacket sta
    #else
        port_rx_receive(exception, rec_state, ETHER_100M);  // get mpacket status
    #endif //RFC1812

    mpacket_received#:                                                         // wait fo

    .if (packet_buf_addr == UNALLOCATED)
        buf_wait();
        #if (FREELIST_ID == 0)
            #define BASE_ADDR  SRAM_BUFF_
        #else
            #define_eval HALF_BUFFER_COU
            #define_eval BASE_ADDR (SRAM_                                  OUNT * 4))
        #endif
        .while ($pop_xfer[0] == BASE_ADDR)
            buf_pop($pop_xfer[0], FREELIST_HANDLE, ctx_swap);                  // if no l
        .endw
        buf_dram_addr_from_sram_addr(packet_buf_addr, $pop_xfer[0], FREELIST_HANDLE);
        move(descriptor_addr, $pop_xfer[0]);
    .endif
    xbuf_free($pop_xfer);
```

IXP1200 Developer Workbench - Error

The simulation is being stopped because the IXP received 100 packets from IX bus.

OK

For Help, select Help->Help Topics on the main menu   uEng/SA: 8532   IX Bus: 3555.00   Ln 69, Col 1


L3fwd16 - IXP1200 Developer Workbench - [C:\IXP1200\MicroCode\threads\rx_ether100m.uc]

```
// main loop
.while (1)

    xbuf_alloc($pop_xfer, 1

    .if (packet_buf_addr ==
        buf_pop($pop_xfer[(
    .endif

    port_rxrdy_chk(@rdready
    critsect_enter[@req_inf
    port_rx_request(rec_rec

    #ifdef RFC1812
        port_rx_receive(exc
    #else
        port_rx_receive(exc
    #endif //RFC1812

    mpacket_received#:

    .if (packet_buf_addr ==
        buf_wait();
        #if (FREELIST_ID ==
            #define BASE_AI
        #else
            #define_eval HA
            #define_eval BA
        #endif
        .while ($pop_xfer[(
            buf_pop($pop_xf
        .endw
        buf_dram_addr_from_
        move(descriptor_add
    .endif
    xbuf_free($pop_xfer);
```

Performance Statistics

Summary | Microengine | All

Statistics were gathered starting at cycle 1.

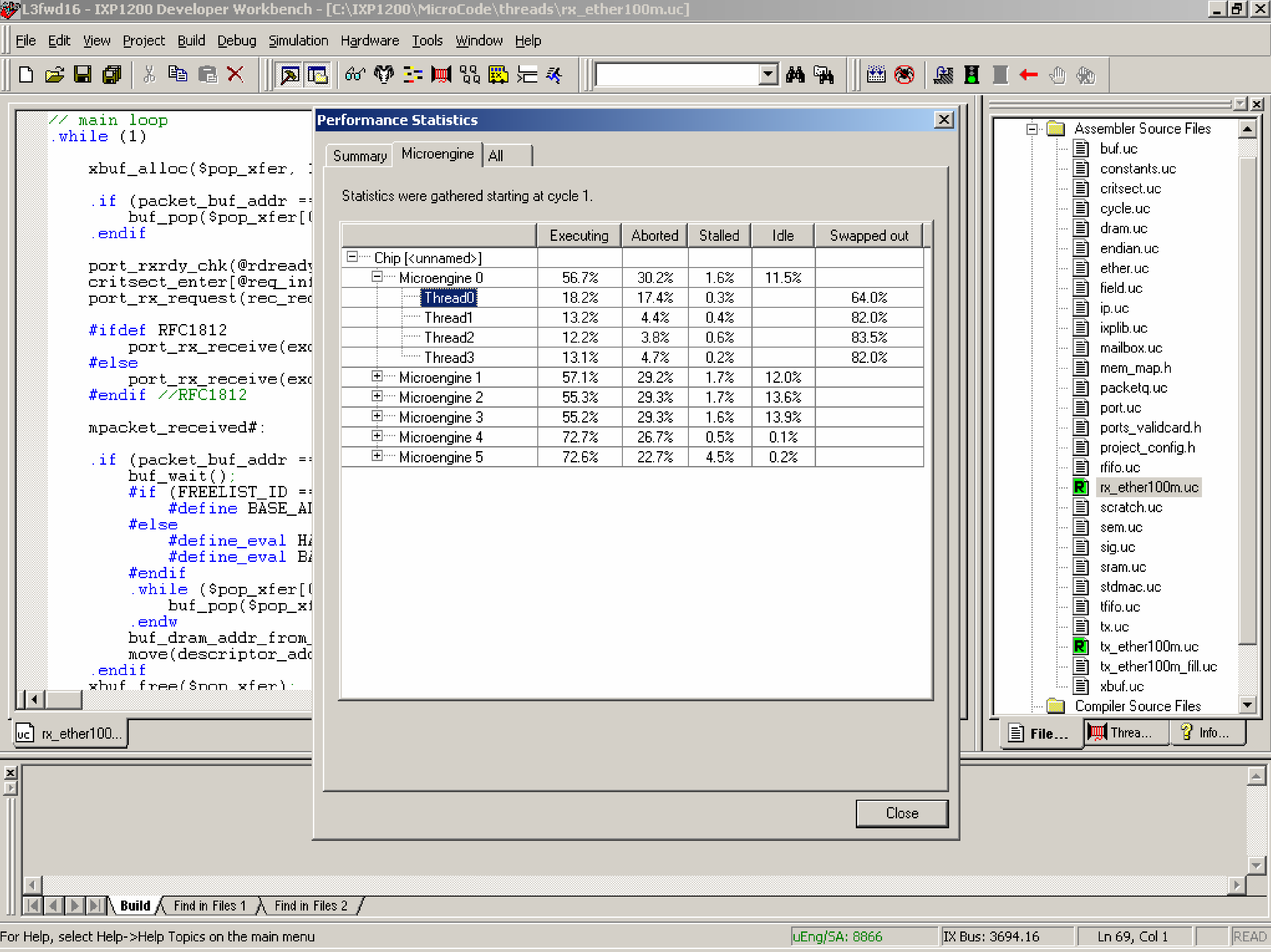
	Active	Rate
Chip [<unnamed>]		
Microengine 0	56.7%	113.4 Mips
Microengine 1	57.1%	114.2 Mips
Microengine 2	55.3%	110.7 Mips
Microengine 3	55.2%	110.5 Mips
Microengine 4	72.7%	145.4 Mips
Microengine 5	72.6%	145.2 Mips
Total		739.3 Mips
SDRAM	39.1%	2499.9 Mb/s
SRAM	46.1%	1476.2 Mb/s

Assembler Source Files: buf.uc, constants.uc, critsect.uc, cycle.uc, dram.uc, endian.uc, ether.uc, field.uc, ip.uc, ixplib.uc, mailbox.uc, mem_map.h, packetq.uc, port.uc, ports_validcard.h, project_config.h, rfifo.uc, rx_ether100m.uc, scratch.uc, sem.uc, sig.uc, sram.uc, stdmac.uc, tfifo.uc, tx.uc, tx_ether100m.uc, tx_ether100m_fill.uc, xbuf.uc

Compiler Source Files

For Help, select Help->Help Topics on the main menu | uEng/SA: 8866 | IX Bus: 3694.16 | Ln 69, Col 1 | READ

File Edit View Project Build Debug Simulation Hardware Tools Window Help
Close
rx_ether100...
File... Threa... Info...
Build Find in Files 1 Find in Files 2


L3fwd16 - IXP1200 Developer Workbench - [C:\IXP1200\MicroCode\threads\rx_ether100m.uc]
File Edit View Project Build Debug Simulation Hardware Tools Window Help
Performance Statistics
Summary Microengine All
Close
Assembler Source Files
Compiler Source Files
rx_ether100...
File... Threa... Info...
Build Find in Files 1 Find in Files 2

```
// main loop
.while (1)

    xbuf_alloc($pop_xfer, 1

    .if (packet_buf_addr ==
        buf_pop($pop_xfer[(
    .endif

    port_rxrdy_chk(@rdready
    critsect_enter[@req_inl
    port_rx_request(rec_rec

    #ifdef RFC1812
        port_rx_receive(exc
    #else
        port_rx_receive(exc
    #endif //RFC1812

    mpacket_received#:

    .if (packet_buf_addr ==
        buf_wait();
        #if (FREELIST_ID ==
            #define BASE_AI
        #else
            #define_eval HA
            #define_eval BA
        #endif
        .while ($pop_xfer[(
            buf_pop($pop_x
        .endw
        buf_dram_addr_from_
        move(descriptor_adc
    .endif
    xbuf_free($pop_xfer);
```

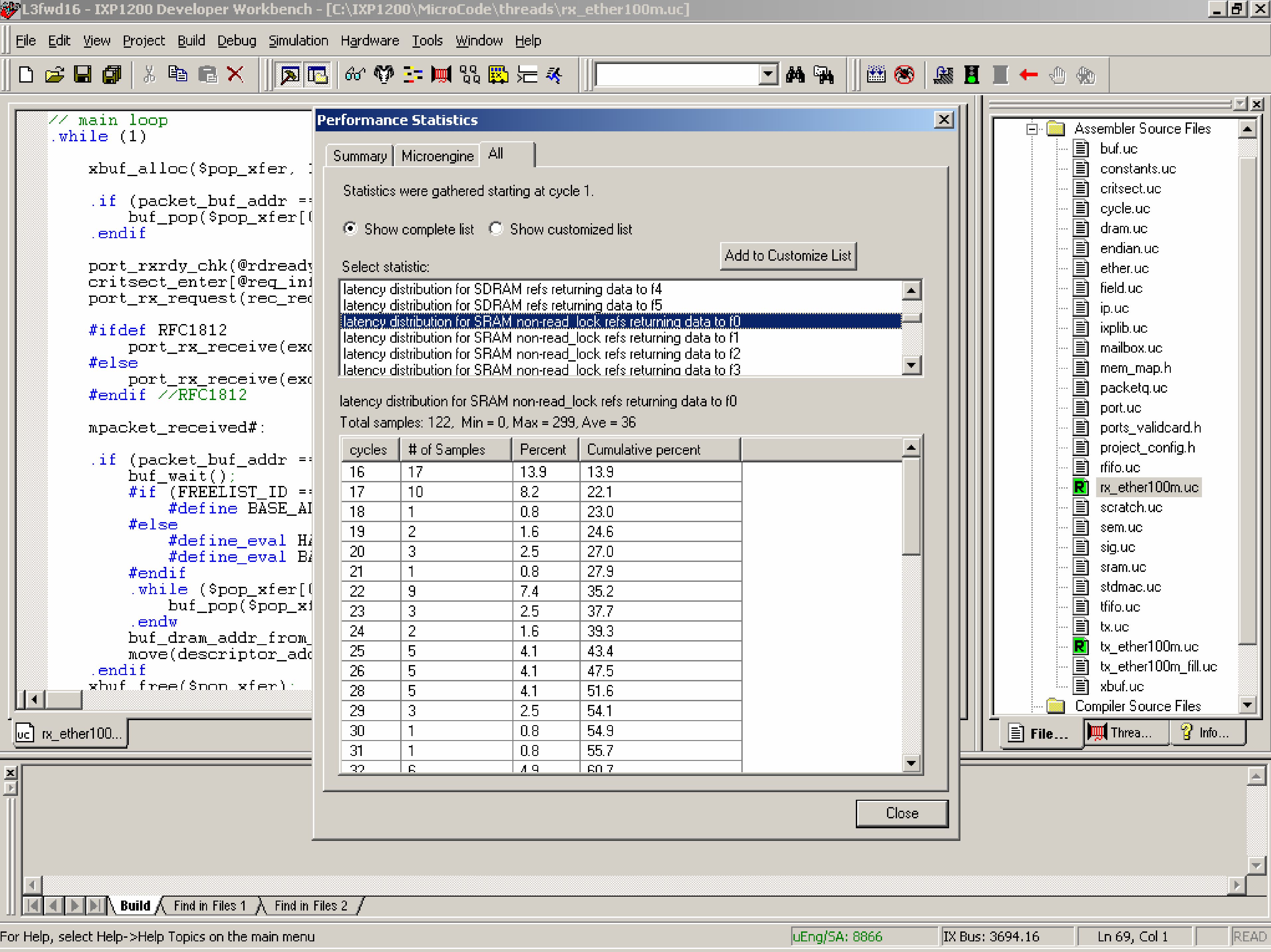
Statistics were gathered starting at cycle 1.

	Executing	Aborted	Stalled	Idle	Swapped out
⊟ Chip [<unnamed>]					
⊟ Microengine 0	56.7%	30.2%	1.6%	11.5%	
Thread0	18.2%	17.4%	0.3%		64.0%
Thread1	13.2%	4.4%	0.4%		82.0%
Thread2	12.2%	3.8%	0.6%		83.5%
Thread3	13.1%	4.7%	0.2%		82.0%
⊞ Microengine 1	57.1%	29.2%	1.7%	12.0%	
⊞ Microengine 2	55.3%	29.3%	1.7%	13.6%	
⊞ Microengine 3	55.2%	29.3%	1.6%	13.9%	
⊞ Microengine 4	72.7%	26.7%	0.5%	0.1%	
⊞ Microengine 5	72.6%	22.7%	4.5%	0.2%	

Assembler Source Files
- buf.uc
- constants.uc
- critsect.uc
- cycle.uc
- dram.uc
- endian.uc
- ether.uc
- field.uc
- ip.uc
- ixplib.uc
- mailbox.uc
- mem_map.h
- packetq.uc
- port.uc
- ports_validcard.h
- project_config.h
- rfifo.uc
- rx_ether100m.uc
- scratch.uc
- sem.uc
- sig.uc
- sram.uc
- stdmac.uc
- tfifo.uc
- tx.uc
- tx_ether100m.uc
- tx_ether100m_fill.uc
- xbuf.uc

Compiler Source Files

For Help, select Help->Help Topics on the main menu | uEng/SA: 8866 | IX Bus: 3694.16 | Ln 69, Col 1 | READ


L3fwd16 - IXP1200 Developer Workbench - [C:\IXP1200\MicroCode\threads\rx_ether100m.uc]
File Edit View Project Build Debug Simulation Hardware Tools Window Help
rx_ether100...
Build Find in Files 1 Find in Files 2
File... Threa... Info...

```
// main loop
.while (1)

    xbuf_alloc($pop_xfer, 

    .if (packet_buf_addr ==
        buf_pop($pop_xfer[
    .endif

    port_rxrdy_chk(@rdready
    critsect_enter[@req_in
    port_rx_request(rec_re

    #ifdef RFC1812
        port_rx_receive(ex
    #else
        port_rx_receive(ex
    #endif //RFC1812

    mpacket_received#:

    .if (packet_buf_addr ==
        buf_wait();
        #if (FREELIST_ID ==
            #define BASE_A
        #else
            #define_eval H
            #define_eval B
        #endif
        .while ($pop_xfer[
            buf_pop($pop_x
        .endw
        buf_dram_addr_from_
        move(descriptor_ad
    .endif
    xbuf_free($pop_xfer);
```

Performance Statistics

Summary | Microengine | All

Statistics were gathered starting at cycle 1.

[✓ Show complete list] Show customized list

Add to Customize List

Select statistic:

- latency distribution for SDRAM refs returning data to f4
- latency distribution for SDRAM refs returning data to f5
- latency distribution for SRAM non-read_lock refs returning data to f0
- latency distribution for SRAM non-read_lock refs returning data to f1
- latency distribution for SRAM non-read_lock refs returning data to f2
- latency distribution for SRAM non-read_lock refs returning data to f3

latency distribution for SRAM non-read_lock refs returning data to f0

Total samples: 122, Min = 0, Max = 299, Ave = 36

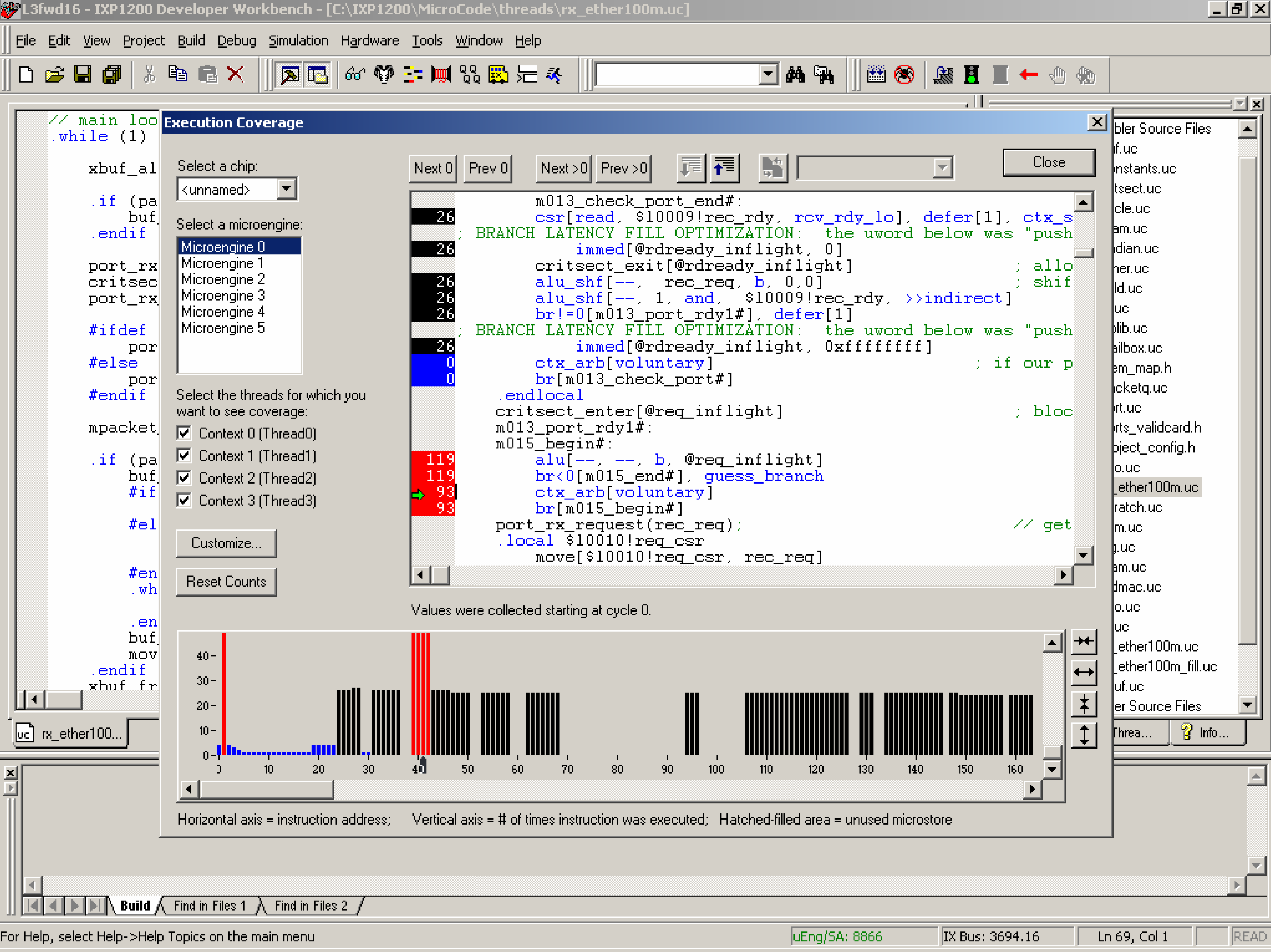
cycles	# of Samples	Percent	Cumulative percent
16	17	13.9	13.9
17	10	8.2	22.1
18	1	0.8	23.0
19	2	1.6	24.6
20	3	2.5	27.0
21	1	0.8	27.9
22	9	7.4	35.2
23	3	2.5	37.7
24	2	1.6	39.3
25	5	4.1	43.4
26	5	4.1	47.5
28	5	4.1	51.6
29	3	2.5	54.1
30	1	0.8	54.9
31	1	0.8	55.7
32	6	4.9	60.7

Close

Assembler Source Files
- buf.uc
- constants.uc
- critsect.uc
- cycle.uc
- dram.uc
- endian.uc
- ether.uc
- field.uc
- ip.uc
- ixplib.uc
- mailbox.uc
- mem_map.h
- packetq.uc
- port.uc
- ports_validcard.h
- project_config.h
- rfifo.uc
- rx_ether100m.uc
- scratch.uc
- sem.uc
- sig.uc
- sram.uc
- stdmac.uc
- tfifo.uc
- tx.uc
- tx_ether100m.uc
- tx_ether100m_fill.uc
- xbuf.uc

Compiler Source Files

For Help, select Help->Help Topics on the main menu | uEng/SA: 8866 | IX Bus: 3694.16 | Ln 69, Col 1 | READ


L3fwd16 - IXP1200 Developer Workbench - [C:\IXP1200\MicroCode\threads\rx_ether100m.uc]
File Edit View Project Build Debug Simulation Hardware Tools Window Help
Execution Coverage
Next 0 Prev 0 Next >0 Prev >0
Close
Customize...
Reset Counts
Build Find in Files 1 Find in Files 2
rx_ether100...
Threa... Info...

Select a chip: <unnamed>

Select a microengine:
Microengine 0
Microengine 1
Microengine 2
Microengine 3
Microengine 4
Microengine 5

Select the threads for which you want to see coverage:
[✓] Context 0 (Thread0)
[✓] Context 1 (Thread1)
[✓] Context 2 (Thread2)
[✓] Context 3 (Thread3)

```
         m013_check_port_end#:
 26          csr[read, $10009!rec_rdy, rcv_rdy_lo], defer[1], ctx_s
   ; BRANCH LATENCY FILL OPTIMIZATION:  the uword below was "push
 26              immed[@rdready_inflight, 0]
             critsect_exit[@rdready_inflight]               ; allo
 26          alu_shf[--,  rec_req, b, 0,0]                  ; shif
 26          alu_shf[--, 1, and,  $10009!rec_rdy, >>indirect]
 26          br!=0[m013_port_rdy1#], defer[1]
   ; BRANCH LATENCY FILL OPTIMIZATION:  the uword below was "push
 26              immed[@rdready_inflight, 0xffffffff]
  0          ctx_arb[voluntary]                          ; if our p
  0          br[m013_check_port#]
         .endlocal
         critsect_enter[@req_inflight]                      ; bloc
         m013_port_rdy1#:
         m015_begin#:
119          alu[--, --, b, @req_inflight]
119          br<0[m015_end#], guess_branch
 93          ctx_arb[voluntary]
 93          br[m015_begin#]
         port_rx_request(rec_req);                          // get
         .local $10010!req_csr
             move[$10010!req_csr, rec_req]
```

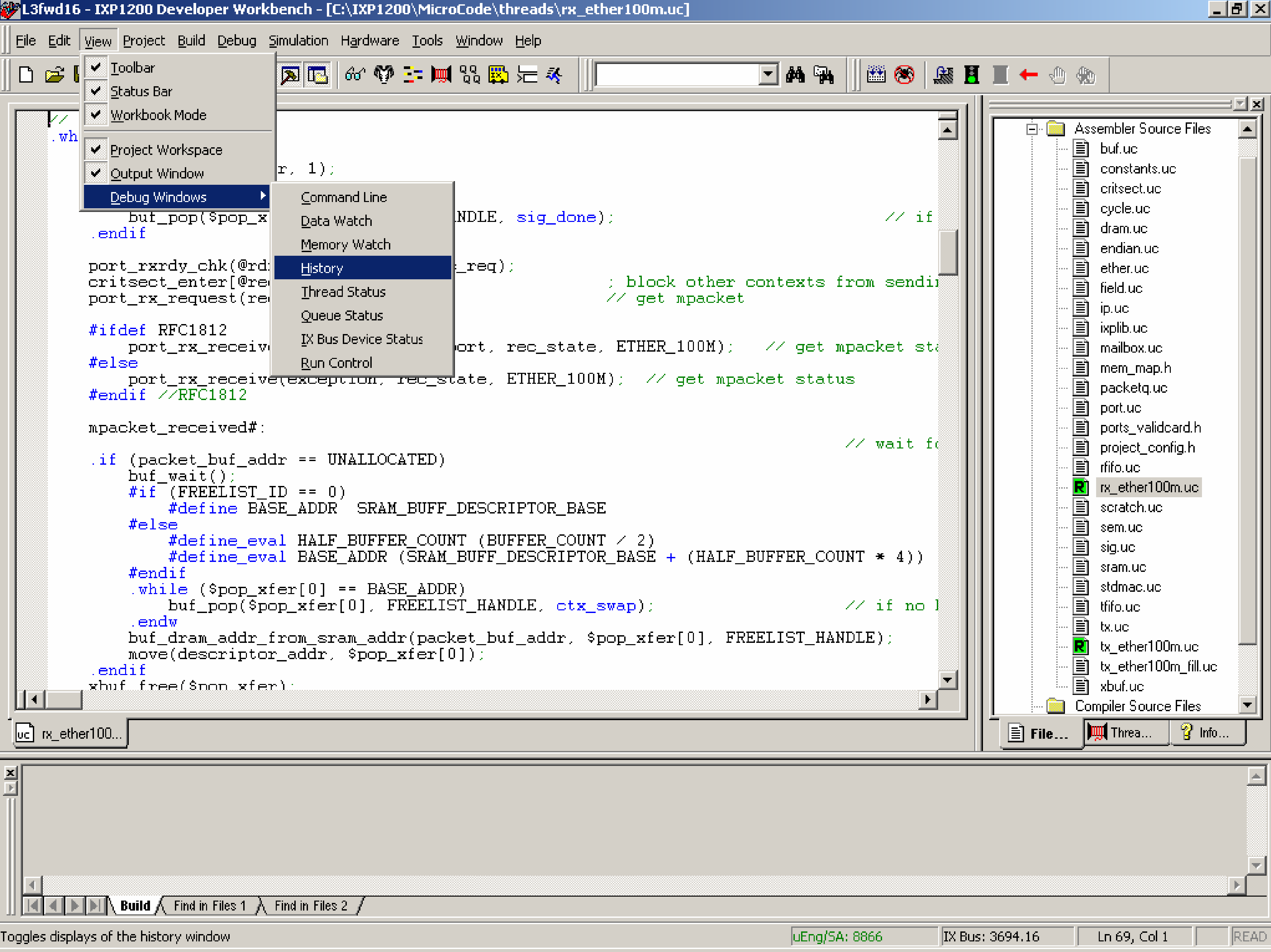
```
// main loo
.while (1)
    xbuf_al
    .if (pa
        buf
    .endif
    port_rx
    critsec
    port_rx
    #ifdef
        por
    #else
        por
    #endif
    mpacket
    .if (pa
        buf
        #if
        #el
        #en
        .wh
        .en
        buf
        mov
    .endif
    xbuf_fr
```

Values were collected starting at cycle 0.

Horizontal axis = instruction address;   Vertical axis = # of times instruction was executed;   Hatched-filled area = unused microstore

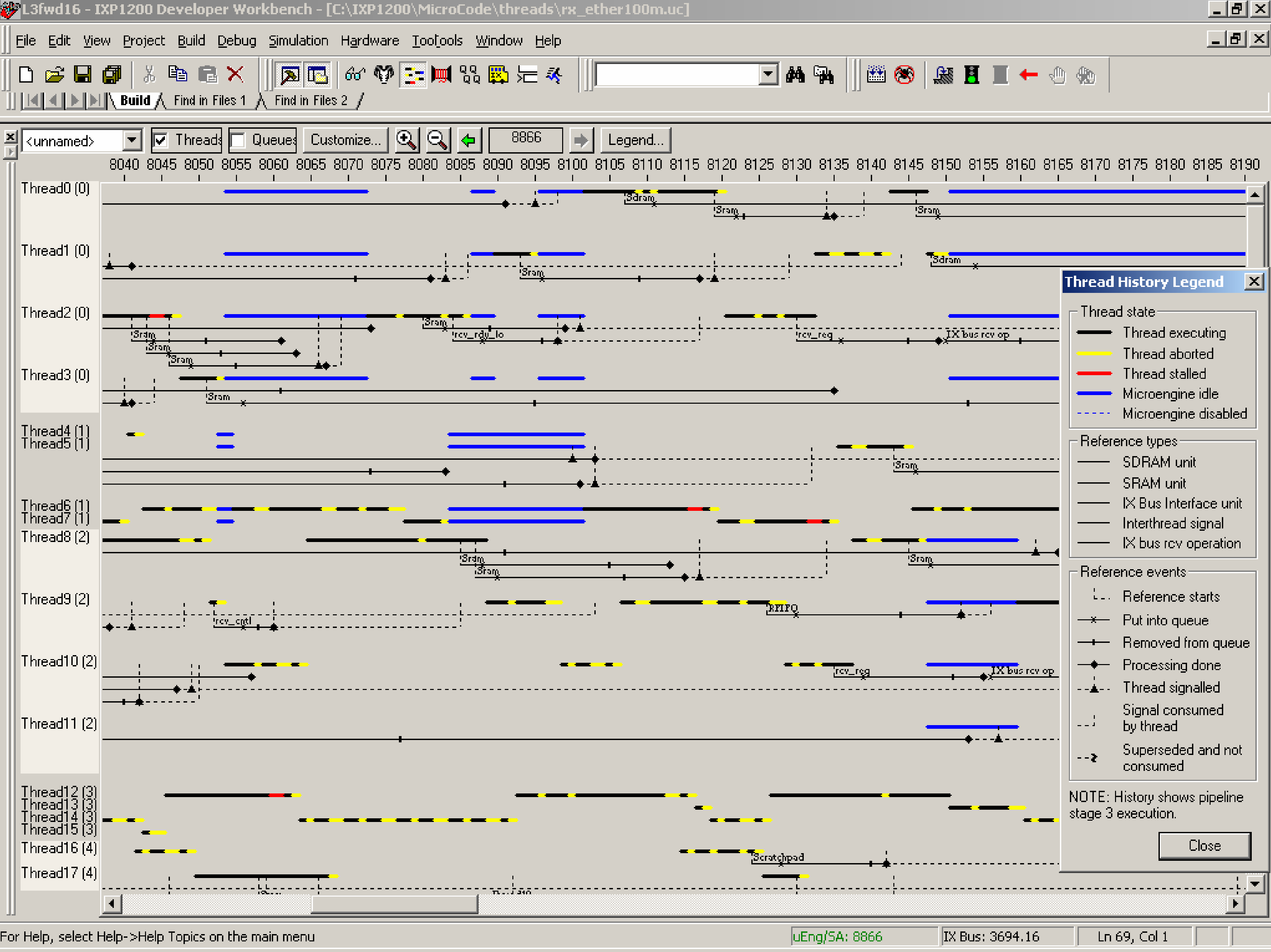
bler Source Files
f.uc
nstants.uc
tsect.uc
cle.uc
am.uc
dian.uc
her.uc
ld.uc
uc
lib.uc
ailbox.uc
em_map.h
cketq.uc
rt.uc
rts_validcard.h
oject_config.h
o.uc
_ether100m.uc
ratch.uc
m.uc
.uc
am.uc
dmac.uc
o.uc
uc
_ether100m.uc
_ether100m_fill.uc
uf.uc
er Source Files

For Help, select Help->Help Topics on the main menu   uEng/SA: 8866   IX Bus: 3694.16   Ln 69, Col 1   READ



```
//
.wh
                                  r, 1);

    buf_pop($pop_x                NDLE, sig_done);                          // if
.endif

port_rxrdy_chk(@rd                _req);
critsect_enter[@re                                ; block other contexts from sendi
port_rx_request(re                                // get mpacket

#ifdef RFC1812
    port_rx_receiv                ort, rec_state, ETHER_100M);   // get mpacket sta
#else
    port_rx_receive(exception, rec_state, ETHER_100M);  // get mpacket status
#endif //RFC1812

mpacket_received#:
                                                                           // wait fo
.if (packet_buf_addr == UNALLOCATED)
    buf_wait();
    #if (FREELIST_ID == 0)
        #define BASE_ADDR  SRAM_BUFF_DESCRIPTOR_BASE
    #else
        #define_eval HALF_BUFFER_COUNT (BUFFER_COUNT / 2)
        #define_eval BASE_ADDR (SRAM_BUFF_DESCRIPTOR_BASE + (HALF_BUFFER_COUNT * 4))
    #endif
    .while ($pop_xfer[0] == BASE_ADDR)
        buf_pop($pop_xfer[0], FREELIST_HANDLE, ctx_swap);                  // if no l
    .endw
    buf_dram_addr_from_sram_addr(packet_buf_addr, $pop_xfer[0], FREELIST_HANDLE);
    move(descriptor_addr, $pop_xfer[0]);
.endif
xbuf_free($pop_xfer);
```

Toggles displays of the history window   uEng/SA: 8866   IX Bus: 3694.16   Ln 69, Col 1   READ



Thread History Legend

Thread state

▬	Thread executing
▬	Thread aborted
▬	Thread stalled
▬	Microengine idle
- - -	Microengine disabled

Reference types

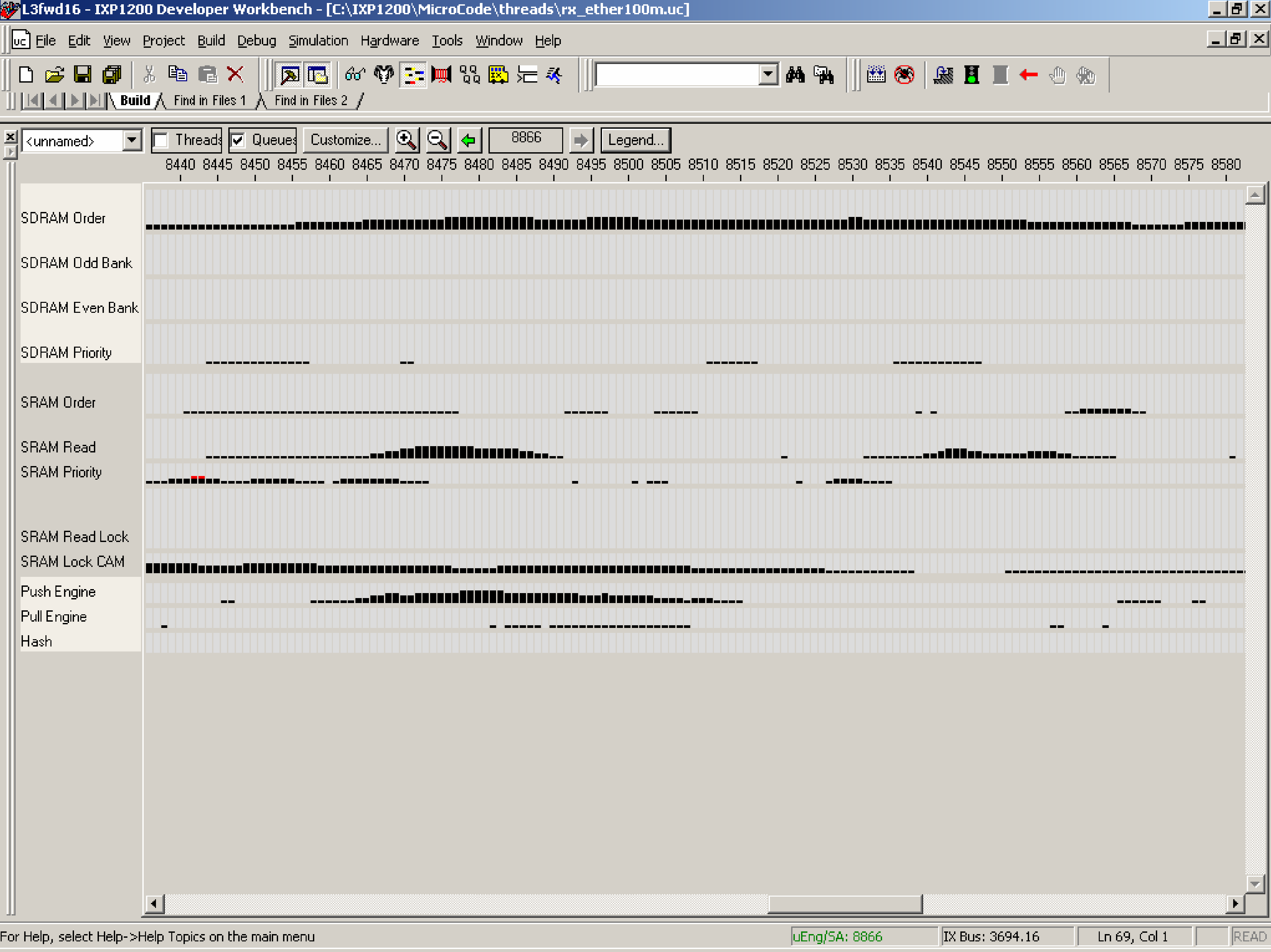
—	SDRAM unit
—	SRAM unit
—	IX Bus Interface unit
—	Interthread signal
—	IX bus rcv operation

Reference events

¦	Reference starts
—×—	Put into queue
—+—	Removed from queue
—◆—	Processing done
-▲-	Thread signalled
-¦	Signal consumed by thread
-‑z	Superseded and not consumed

NOTE: History shows pipeline stage 3 execution.

For Help, select Help->Help Topics on the main menu | uEng/SA: 8866 | IX Bus: 3694.16 | Ln 69, Col 1


L3fwd16 - IXP1200 Developer Workbench - [C:\IXP1200\MicroCode\threads\rx_ether100m.uc]
File Edit View Project Build Debug Simulation Hardware Tools Window Help
Build
Find in Files 1
Find in Files 2
<unnamed>
Threads
Queues
Customize...
8866
Legend...
8440 8445 8450 8455 8460 8465 8470 8475 8480 8485 8490 8495 8500 8505 8510 8515 8520 8525 8530 8535 8540 8545 8550 8555 8560 8565 8570 8575 8580
SDRAM Order
SDRAM Odd Bank
SDRAM Even Bank
SDRAM Priority
SRAM Order
SRAM Read
SRAM Priority
SRAM Read Lock
SRAM Lock CAM
Push Engine
Pull Engine
Hash
For Help, select Help->Help Topics on the main menu
uEng/SA: 8866
IX Bus: 3694.16
Ln 69, Col 1
READ

Lab 2

- Part I: Collect statistics
  - Microengine utilization for all microengines
  - Detailed statistics of one thread from uE 0 and one from uE 5
  - Processing power of microengines (in MIPS).
  - Memory utilization and bandwidth.
  - Latency distribution for SDRAM refs for microengine 0 and SRAM non-read_lock refs for microengine 0. Show a graph.
  - Show a screenshot for the thread history that shows overlapping SRAM and SDRAM requests by the same microengine.
  - Identify the overall delay for either request (in cycles). What factors contributed how much to the overall delay?
- DUE NEXT TUESDAY.

Lab 1 Results

- Grading: 20 points total
  - Results: 10 points
  - Code: 3 points
  - TCP state machine + explaination: 2+1 points
  - IP and TCP headers: 1+1 points
  - Report (written content): 2 points
- Average: 16.6
- Max: 20
- Min: 14

Next Class

- Microengine programming
  - Assembler
  - Instructions
  - Register access
  - Assembler directives
  - Etc.
- Read Chapter 24
- Turn in Part I of Lab 2