June 27, 2007

MPC8568E Table Lookup Unit Development

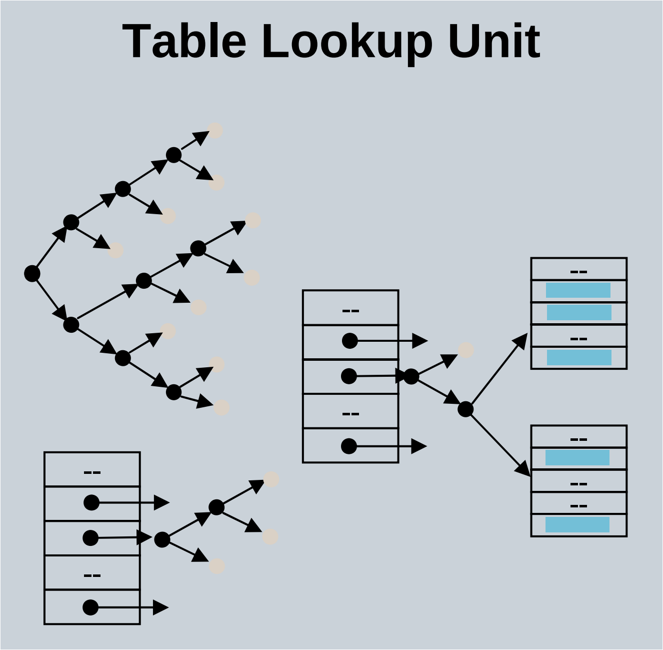
AN304

Sam Siu

System and Application Engineer

Agenda

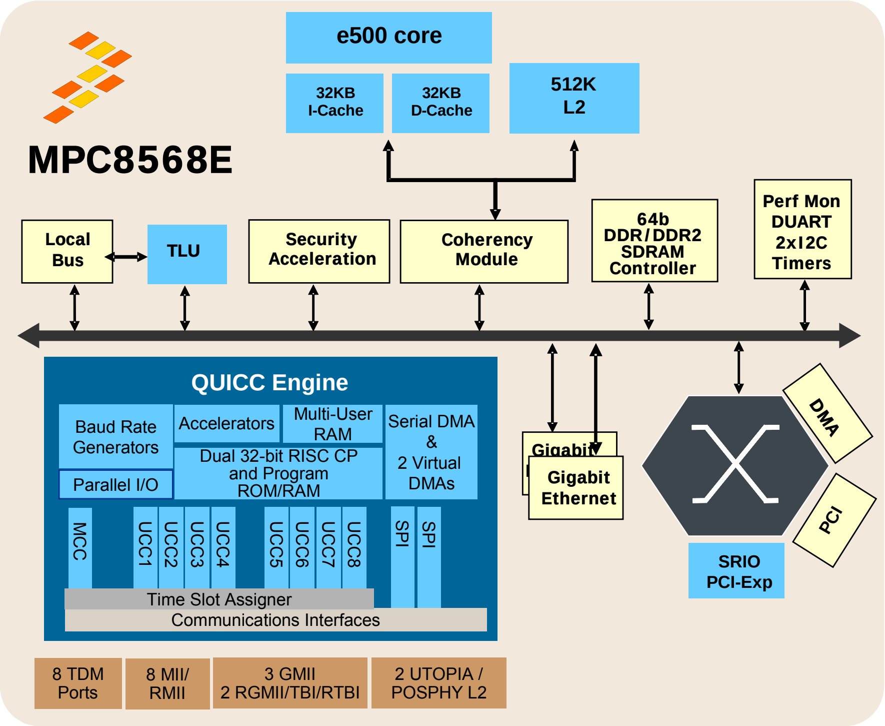
- System Overview
- Table Lookup Unit Hardware Overview
- Table Lookup Unit Software Overview
- Table Lookup Unit Development Environment
- Table Lookup Unit Sample Applications
- Conclusion

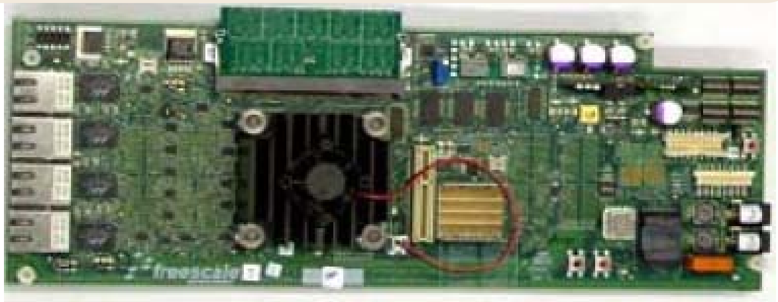


MPC8568E Overview

Key Advantages

- High level of integration simplifying board design
- Consistent programming model across the PowerQUICC® III family of processors
- 90 nm silicon-on-insulator (SOI) technology
- High-performance enhanced e500 core
  - 512 KB L2 cache
  - High internal processing bandwidth
- Integrated DDR/DDR2 memory controller
- 2 * integrated Triple Speed Ethernet Controllers
- Advanced QUICC Engine™ technology supports a wide range of protocols and associated internetworking applications
- TLU provides off-load for table search functions associated with IP forwarding and firewall.
- Flexible high-speed interconnect interfaces:
  - Serial RapidIO® interconnect technology
  - PCI Express ® support
- PCI and local bus interface support
- Integrated security engine





Agenda

- ▶ System Overview
- ▶ Table Lookup Unit Hardware Overview
- ▶ Table Lookup Unit Software Overview
- ▶ Table Lookup Unit Development Environment
- ▶ Table Lookup Unit Sample Applications
- ▶ Conclusion

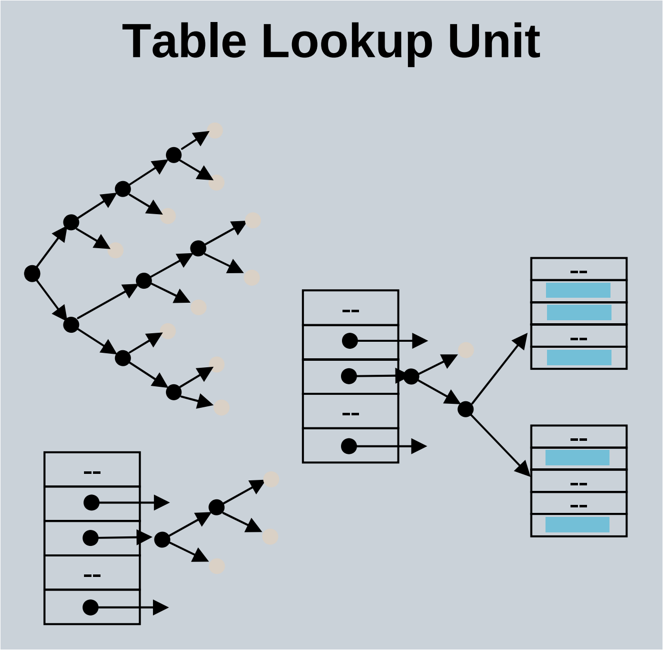
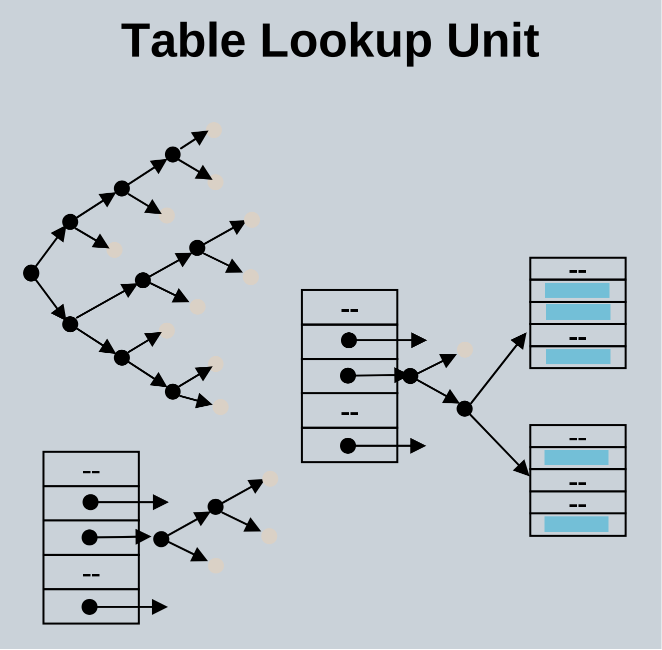


Table Lookup Unit (TLU): An Offload Engine

- TLU is a flexible table lookup offload engine that supports:
  - The exact match
  - Longest prefix match lookups
  - Which is used for IP forwarding and networking applications
- Number of entries and entry size are configurable
  - Max table size is 16M entries or 128MB
  - Max entry size is 64 bytes
- Level performance for large table sizes
  - 5 M lookups/sec
- Support both SRAM and SDRAM
  - 4 memory banks
  - Maximum 256MB per bank
- Integrated real-time statistics counters



TLU Features

- Individually configurable table structures
  - Up to 32 physical tables available, with each table up to 16M entries.
  - 8, 16, 32, and 64-byte table entry sizes supported
- Supports four advance table types
  - Hash-trie-key table for hash-based exact-match algorithms
  - Chained hash table for partially indexed and hashed exact-match algorithms
  - Variable prefix-expansion trie-data table for longest prefix match algorithm
  - Flat data table for retrieving search results and simple indexed algorithms
- Flexible command set
  - Direct table entry reads and writes for safe in-place updates
  - Key find operations on 32, 64, 96, and 128-bit keys, with don’t care bits masked to zero
- High-performance hash capability
  - Hash function minimizes collisions on both real-world data and pathological patterns, and offers superior randomization over cyclic-redundancy codes

TLU Algorithm Overview

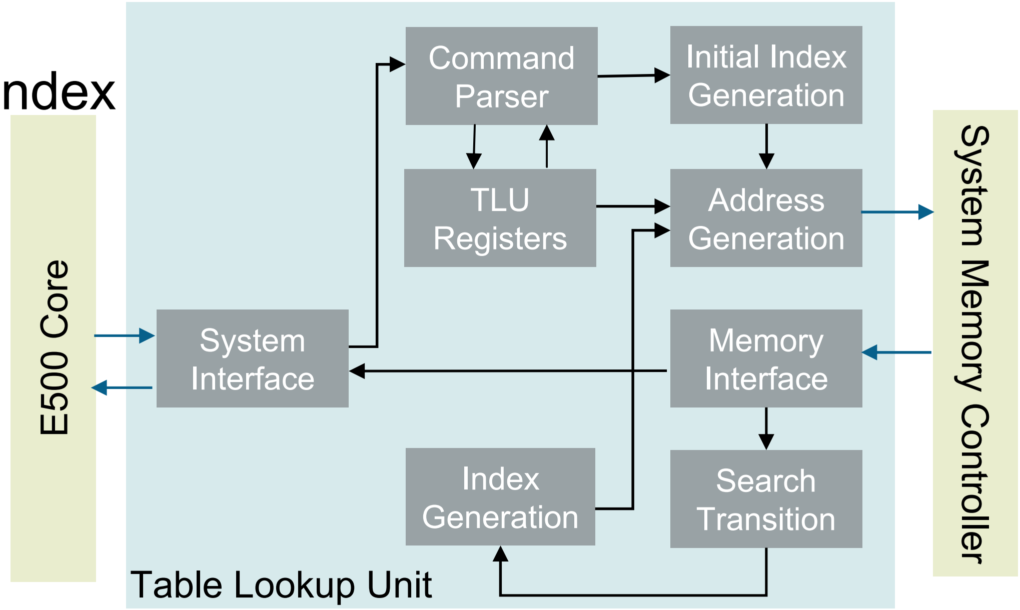
The Table Lookup Unit implements two kinds of lookup algorithm:

- Exact match
  - Useful for flow identification, switching, policy database applications.
  - Flat data tables for simple indexed data retrieval.
  - Hash-Trie-Key (HTK) tables for hash-based lookup of keys from 32 bits to 128 bits with binary Tries for collision resolution.
  - Chained-Hash tables using an index-HTK combination.
- Longest-Prefix match
  - Index-Variable Prefix Trie-Data (IPTD) tables are radix trie data structures designed for performing efficient CIDR IP routing.

Algorithm	Initial Simple Table	Subsequent Tables/States
Flat Data	Data	—
Hash-Trie-Key (HTK)	Hash	Trie, Key, Fail
Chained Hash	IPTD	IPTD, Data, Hash, Trie, Key, Fail
Compressed Radix Trie (CRT)	IPTD	IPTD, Data, Hash, Trie, Key, Fail

Table Lookup Unit Block Diagram

- Dedicated low-latency connection to the local bus controller (LBC)
- Support variety of table lookup algorithms to meet different needs.
  - Parses a TLU command from the CPU
  - Calculates the initial index base on a key
  - Evaluates the current table node
    - Fetches memory data at the current index.
    - Fetches a portion of the Key.
  - Fetches the data at the current index
    or
    Calculate new index
  - Returns the data to the CPU

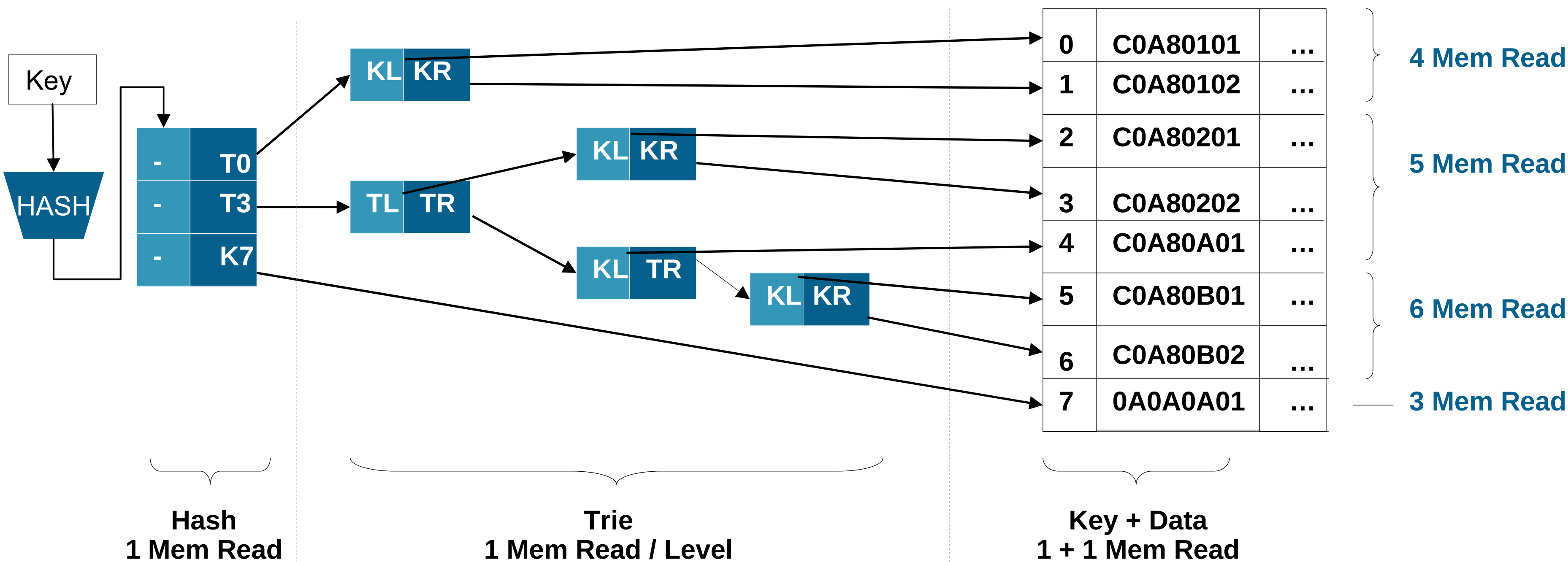


Exact Match Algorithms

- Simple Table type: Flat Data and Chained Hash
- Hash-trie-key Table consist of a hash table, a number of trie entries and a key data table
  - Hash and key tables size are constant
  - Trie depends on number of collisions

|<-Trie Entry Format (32bit) ->|

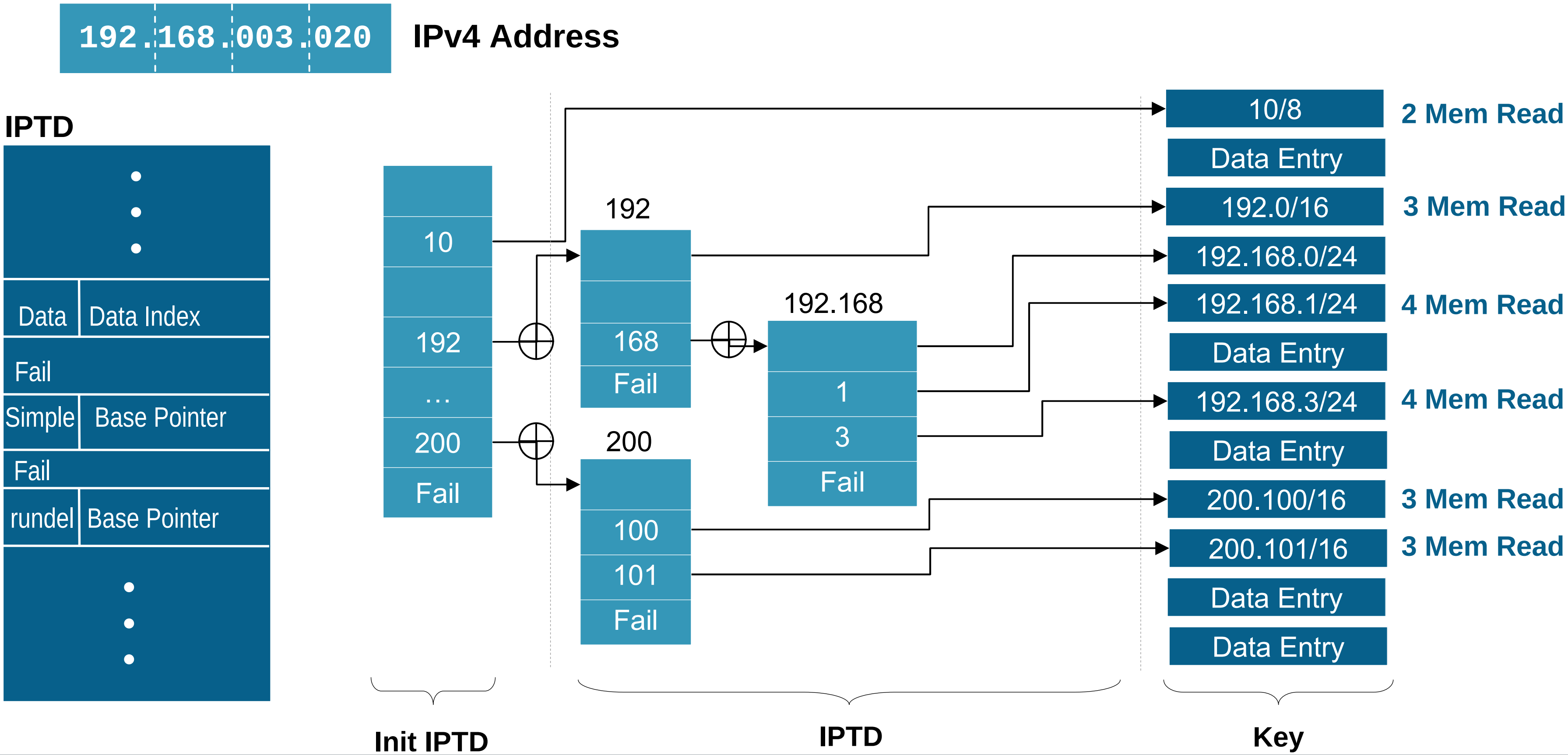
F	Count	L hash entry link ptr
F	Count	R hash entry link ptr



Longest-Prefix Match Algorithms

▶ Compressed Radix Trie (CRT)

- The CRT data structure is formed by linking together an initial IPTD index table, followed by trie sub-tables, possibly compressed.



Memory Consideration

▶ TLU Memory Requirements

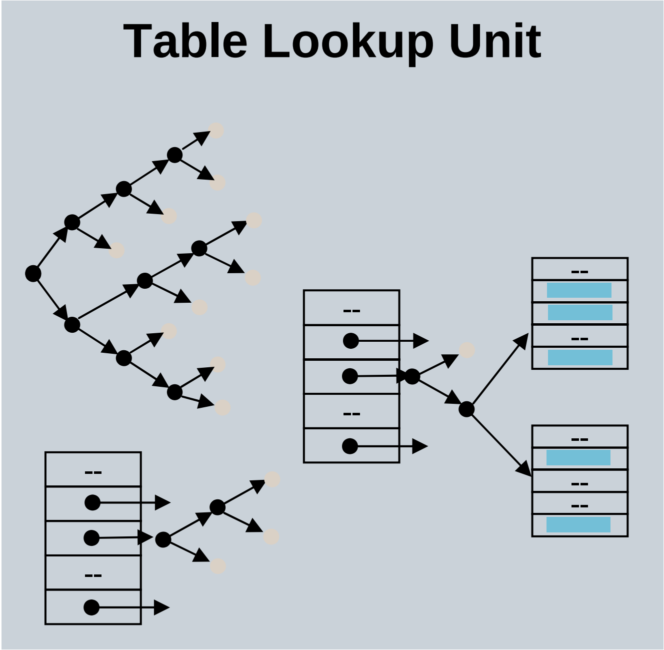
Type	Alignment (B)	Maximum Size (B)
Bank Memory	256M	256M
TLU Table Memory	4K	128M
Initial Hash Table of HTK	Starting at offset 0 of a table memory	256K
Initial IPTD Table of CRT	Starting at offset 0 of a table memory	512K
IPTD Table	8	128M
Trie and Key-Data Table	8	128M

▶ Physical Memory Allocation Approaches

Approaches	Function	Remarks
Slab Allocator	kmalloc	Limit 128K in 2.6
Physical Page Allocator	_get_free_pages	Limit 8M in 2.6 Kernel
Boot Memory Allocator	alloc_bootmem	Need patch kernel. There is an existing patch called bigphys.
Physical Reservation		Linux kernel option “mem=” can reserve high memory.

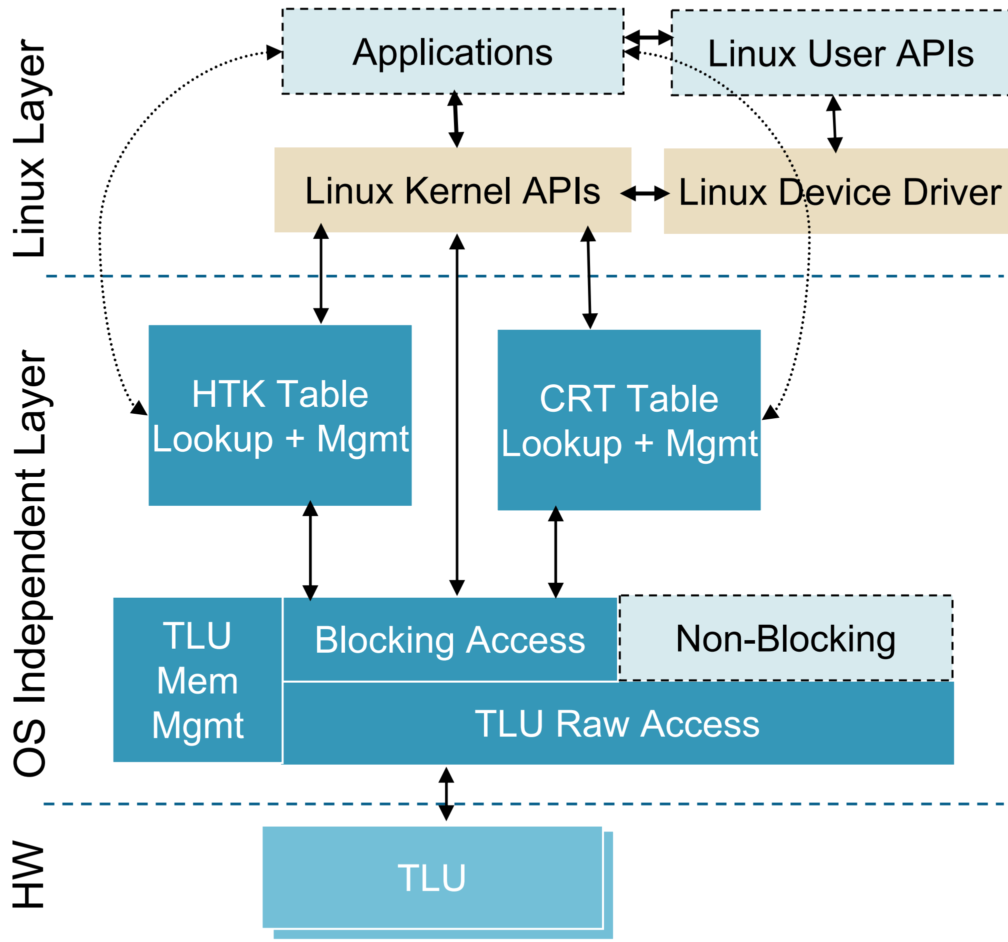
Agenda

- ▶ System Overview
- ▶ Table Lookup Unit Hardware Overview
- ▶ Table Lookup Unit Software Overview
- ▶ Table Lookup Unit Development Environment
- ▶ Table Lookup Unit Sample Applications
- ▶ Conclusion



TLU Software Architecture

- Software driver offer access to TLU hardware
- TLU SW consists of:
  - OS Independent Driver
    - Memory Management
    - HW Raw Access
    - Blocking Mode Access
    - Table Management
  - Linux Kernel APIs
  - Linux Device Driver
  - Directly supported
    - Exact match (HTK)
    - Longest-prefix-match (CRT)



TLU Driver APIs

- Table Services API (C code) to build and access tables
  - Data Structures
    - t_TluHashEntry, t_TluBankParam, t_TluParam, t_TluTableParam, t_TluCrtParam, t_TluHtkParam.
  - Functions
    - TLU_Config(t_TluParam *p_Param, uint32_t baseAddr): create an Hadle for TLU
    - TLU_Init(t_Handle h_Tlu): Initialize TLU
    - TLU_ConfigExceptions(t_Handle h_Tlu, uint8_t imask): Configure TLU Interrupt Mask
  - Run Time
    - TLU_TABLE_WriteWords : writes data to table using double word index
    - TLU_TABLE_Read: reads entry from a TLU table.
    - TLU_TABLE_FindAndRead: search a key in the specified TLU table and read the found data.
    - TLU_TABLE_Create: Create and Initialize a TLU table.
    - t_Handle TLU_CRT_Create: Create and Initialize a TLU CRT table.
    - t_Handle TLU_HTK_Create: Create and Initialize a TLU HTK table.
    - TLU_CRT_Create, TLU_CRT_Free
    - TLU_HTK_Create, TLU_HTK_Free
    - TLU_CRT_Insert, TLU_CRT_Delete, TLU_CRT_Read, TLU_CRT_FindAndRead, TLU_CRT_Write
    - TLU_HTK_Insert, TLU_HTK_Delete, TLU_HTK_Read, TLU_HTK_FindAndRead, TLU_HTK_Write

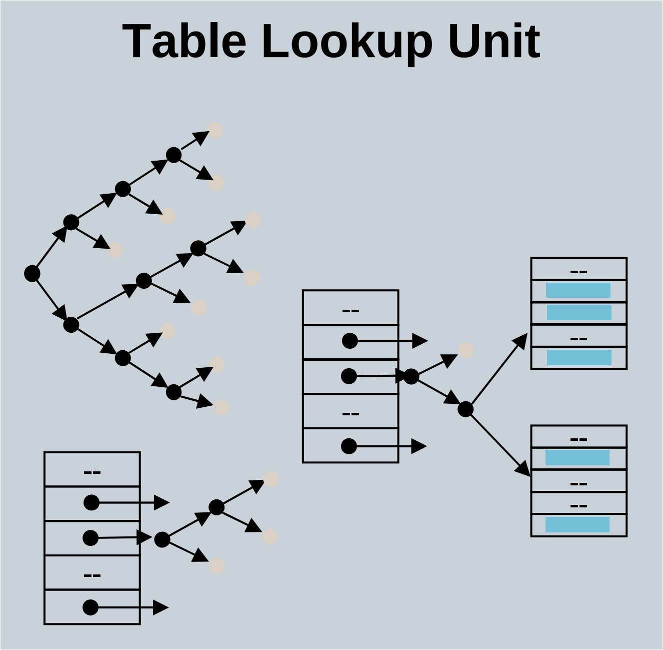
TLU Driver Configuration

- /etc/tlu.conf

```
#Initial log level. 0--off 0x00000010--Access 0x00000020--HTK 0x00000040—CRT log_level=0x03
#Physical address of TLU's processor interface
#Address in 8568
tlu0_addr=0xE002F000
#Physical address of bank 0. Value 0 instructs the driver to dynamically allocate the mem.
tlu0_bank0_addr=0
#Bank memory size in byte. Value 0 indicates the bank is not present.
tlu0_bank0_size=0x00400000
#Parity enable: 1 enables parity check and 0 disables parity check
tlu0_bank0_parity=0
#Bank memory type: 0--Local Bus  1--System DDR
tlu0_bank0_type=1
...
tlu0_bank3_addr=0
tlu0_bank3_size=0
tlu0_bank3_parity=0
tlu0_bank3_type=0
...
### TLU1 configuration
tlu1_addr=0
tlu1_bank0_addr=0
...
tlu1_bank3_addr=0
tlu1_bank3_size=0
tlu1_bank3_parity=0
tlu1_bank3_type=0
```

Agenda

- System Overview
- Table Lookup Unit Hardware Overview
- Table Lookup Unit Software Overview
- Table Lookup Unit Development Environment
- Table Lookup Unit Sample Applications
- Conclusion



MPC8568E MDS Processor Board

1. MPC8568E MDS Processor Board
   - CPU: 990 MHz
   - CCB: 396 MHz
   - DDR2: 198 MHz
2. CodeWarrior® USB Tap
3. Two RS232 port for UART1/2
4. 4 GbE cables
5. 2 GbE loopback cables
6. Documentation
   - Readme
   - Reference Manual
   - Linux® kernel 2.6.20
7. Power supply kit (not shown)

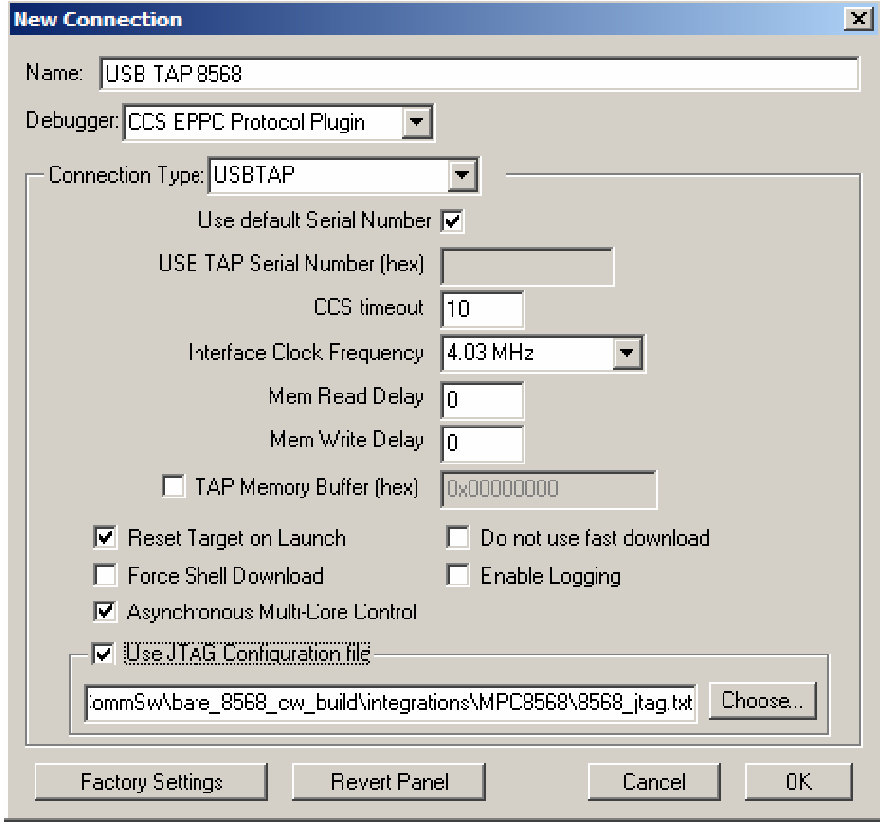
Linux BSP for MPC8568E

BSP Features:

- Tool Chain Version
  - gcc3.4-e500, glibc2.3.4, binutils 2.15 supporting the DPFP of e500v2 core
- Linux® 2.6.20 kernel supporting the e500 v2 core
  - LTIB integration
  - E500 hardware floating point exception handler patches to support the scalar SPFP, vector SPFP and DPFP
  - eTSEC1& eTSEC2 driver to support 10M/100M/1000M Ethernet functionality
  - TCP/IP stack, FTP client and server, Telnet client and server, Web server (boa)
  - Both NFS and Ramdisk filesystems supported
- Bootloader
  - U-Boot 1.1.6
  - Boot from Flash
  - E500 v2 core initialization
  - DDR2 SDRAM initialization
  - Flash Read/Write operation
  - Single serial port at 115200 Baud without flow control
  - eTSEC operation supporting TFTP
  - Load kernel and file system images from Flash
  - I^2C driver to read SPD information from the DDR2 DIMM

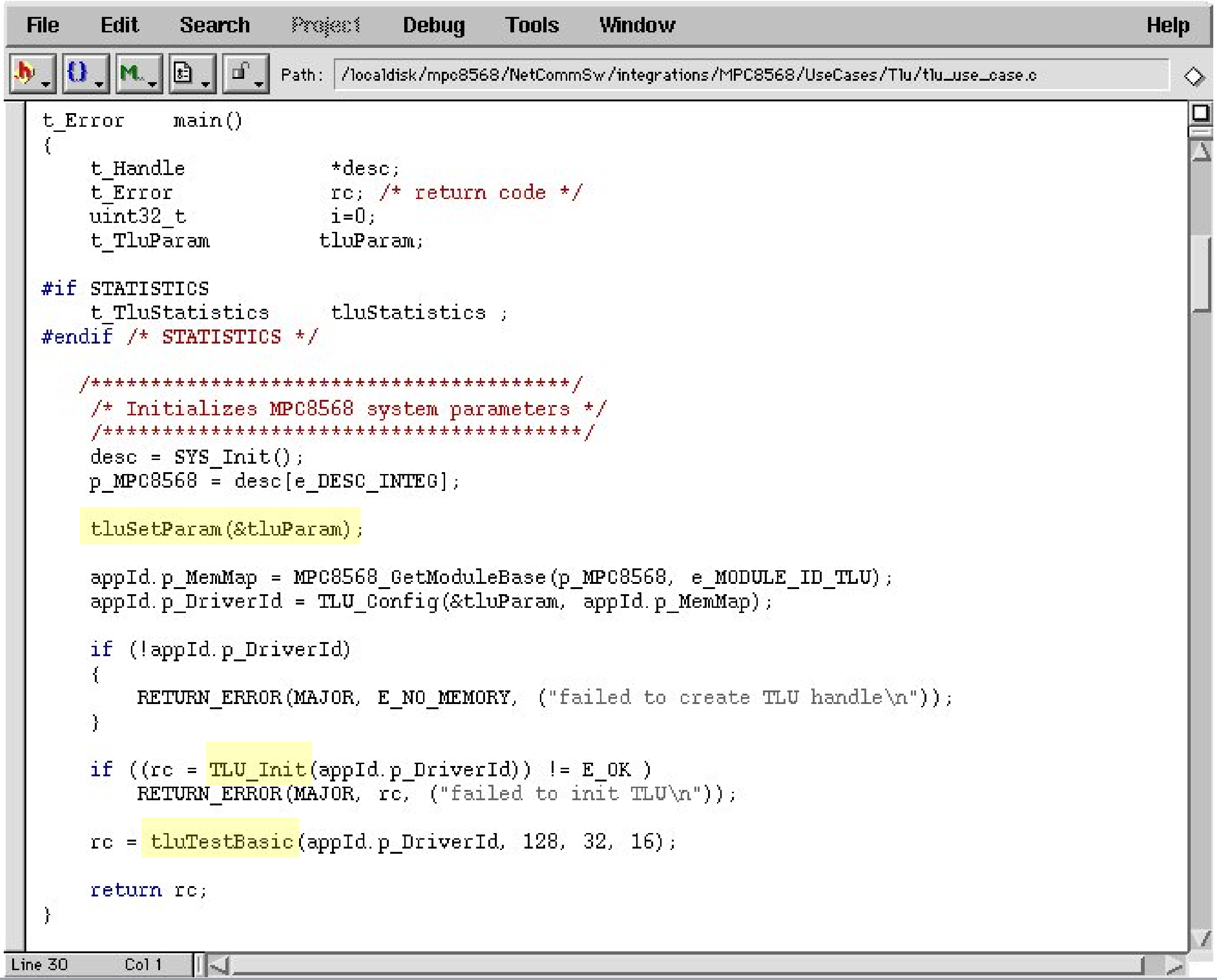
CodeWarrior® USB TAP Interface

- CodeWarrior® Development Studio v8.7, Power Architecture™ technology
- MPC8568E Remote Connection profile



CodeWarrior® Development Environment

- ▶ Sample files
  - tluUseCase.mcp
  - tlu.c
  - tlu_use_case.c
    - Set parameters
    - TLU_Init
    - tluTestBasic



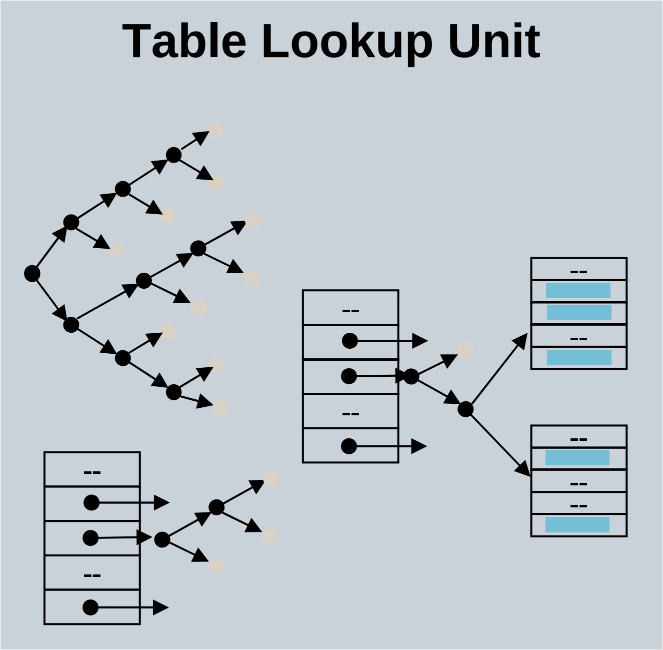

```c
t_Error    main()
{
    t_Handle            *desc;
    t_Error             rc; /* return code */
    uint32_t            i=0;
    t_TluParam         tluParam;

#if STATISTICS
    t_TluStatistics     tluStatistics ;
#endif /* STATISTICS */

   /******************************************/
    /* Initializes MPC8568 system parameters */
    /*****************************************/
    desc = SYS_Init();
    p_MPC8568 = desc[e_DESC_INTEG];

    tluSetParam(&tluParam);

    appId.p_MemMap = MPC8568_GetModuleBase(p_MPC8568, e_MODULE_ID_TLU);
    appId.p_DriverId = TLU_Config(&tluParam, appId.p_MemMap);

    if (!appId.p_DriverId)
    {
        RETURN_ERROR(MAJOR, E_NO_MEMORY, ("failed to create TLU handle\n"));
    }

    if ((rc = TLU_Init(appId.p_DriverId)) != E_OK )
        RETURN_ERROR(MAJOR, rc, ("failed to init TLU\n"));

    rc = tluTestBasic(appId.p_DriverId, 128, 32, 16);

    return rc;
}
```

Linux Kernel APIs

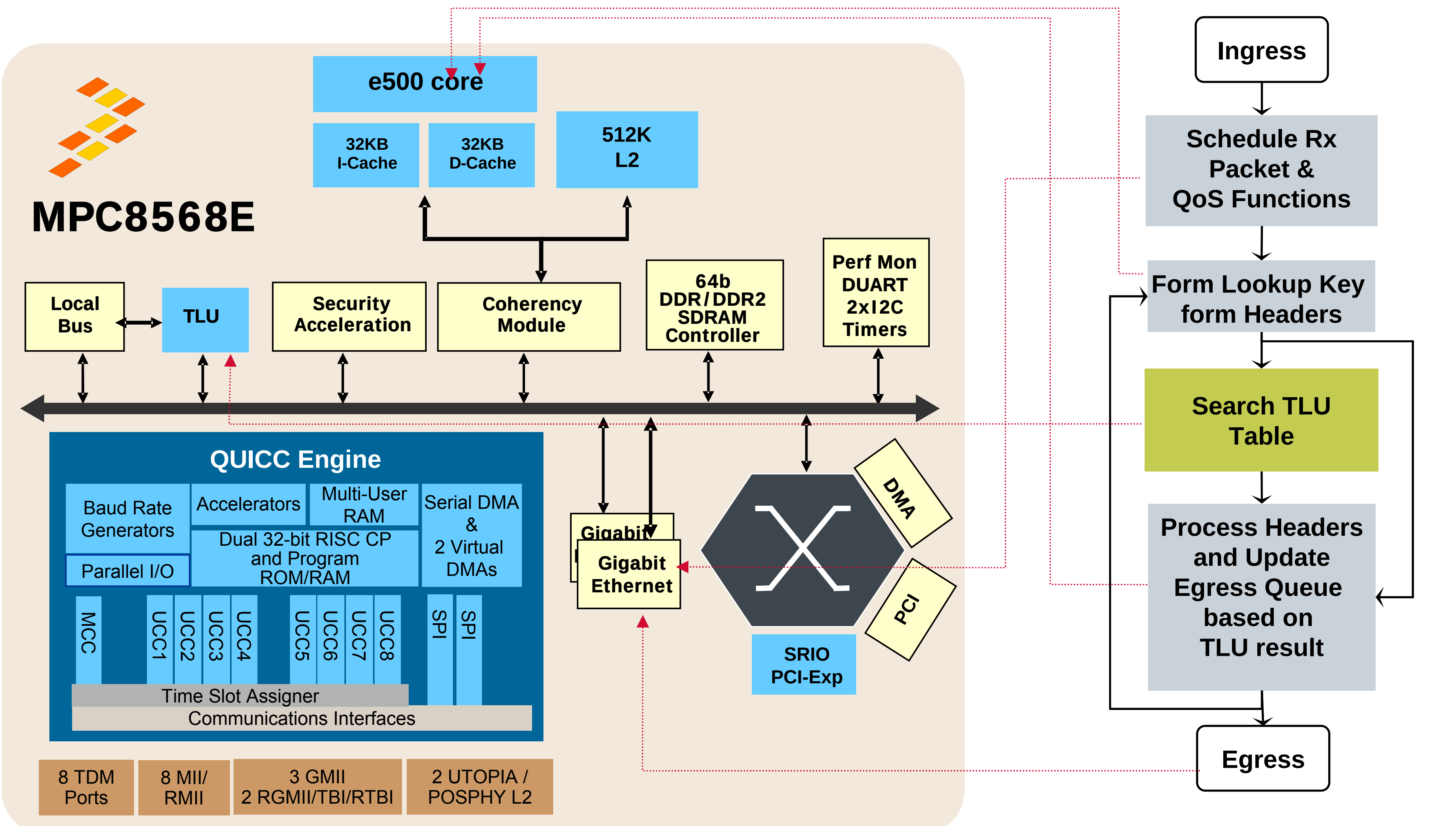
- Device driver APIs
  - tlu_get – Gets the data structure of a TLU for future access
- HTK Table management APIs
  - tlu_htk_create – Creates an HTK table
  - tlu_htk_free – Frees an HTK table.
  - tlu_htk_insert – Inserts a key-data entry into an HTK table.
  - tlu_htk_delete – Deletes a key-data entry from an HTK table.
  - tlu_htk_find – Lookup a key in an HTK table.
  - tlu_htk_findr – Lookup a key in an HTK table and read data if found.
  - tlu_htk_findw – Lookup a key in an HTK table and write the entry.
- CRT Table management APIs
  - tlu_crt_create – Creates a CRT table
  - tlu_crt_free – Frees a CRT table.
  - tlu_crt_insert – Inserts a data entry into a CRT table.
  - tlu_crt_delete – Deletes an entry from a CRT table.
  - tlu_crt_find – Lookup a key in a CRT table.
  - tlu_crt_findr – Lookup a key in a CRT table and read data if found.
  - tlu_crt_findw – Lookup a key in a CRT table and write the entry.

Agenda

- System Overview
- Table Lookup Unit Hardware Overview
- Table Lookup Unit Software Overview
- Table Lookup Unit Development Environment
- Table Lookup Unit Sample Applications
- Conclusion

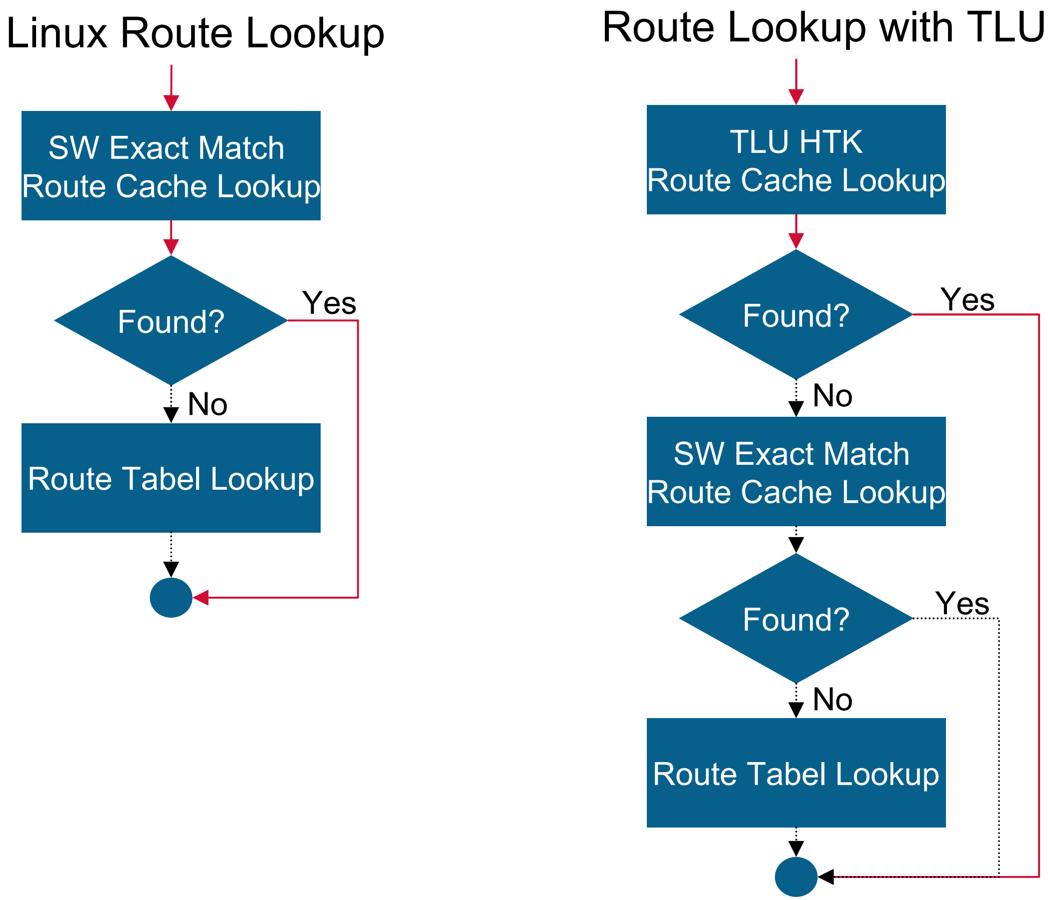


Typical Forwarding Task Sequence



Linux IP Routing Application

- IP Routing can be off loaded to TLU Exact Match lookup

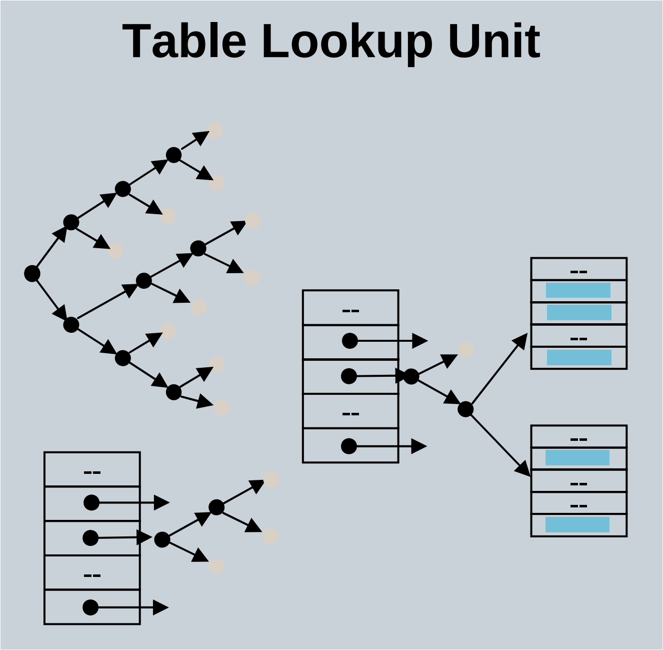


Coding Example: tlu_route.c

```
static inline void* tlu_route_cache_lookup(short iif, short oif, uint32_t daddr,
                uint32_t saddr, int tos)
{
        int rc;
        struct tlu_route_cache entry;

        entry.iif = iif;
        entry.oif = oif;
        entry.daddr = daddr;
        entry.saddr = saddr;
        entry.tos = tos;

#ifdef ROUTE_CACHE_STATS
        _route_cache_stats.lookup_count++;
#endif
#ifdef ROUTE_CACHE_DEBUG
        print_memory(entry.key, ROUTE_CACHE_TABLE_ENTRY_SIZE,
                        entry.key, "LOOK ");
#endif
        if ((rc = _tlu_findr(tlu_addr, 0, entry.key,
                                        ROUTE_CACHE_TABLE_KEY_BYTES,
                                        ROUTE_CACHE_TABLE_KEY_BYTES,
                                        TLU_UNIT_SIZE, &entry.ctx)) < 0){
                if (rc != TLU_NOT_FOUND) {
                        printk("TLU find error %d\n", rc);
                }
                return NULL;
        }
        return (void*)1;
}
                                                                194,0-1         80%
```

Agenda

- ▶ System Overview
- ▶ Table Lookup Unit Hardware Overview
- ▶ Table Lookup Unit Software Overview
- ▶ Table Lookup Unit Development Environment
- ▶ Table Lookup Unit Sample Applications
- ▶ Conclusion



Conclusion

- Compatible superset of C-5e TLU specification
  - Supports new hash function
- Changed memory interface
  - Uses PowerQUICC® III Local Bus memory controller to access ZBT SRAM or SDRAM
- Changed platform interface from C-5e
  - Dedicated TLU service bus provides low-latency access to e500 core
- Variety of table types and keys
  - 32, 48, 96, and 128-bit keys
  - Longest prefix match, chained, hash and flat data table formats
- C base APIs to manage TLU

TLU Related Collateral

- Documentation
  - MPC8568E Reference Manual
  - MPC8572E Reference Manual
  - MPC8568E MDS Processor Board, HW Getting Started Guide
- Development System
  - MPC8568MDS
- Software
  - SDK
  - Linux® 2.6.20 kernel, gcc3.4-e500
- Application Notes
  - AN2755: SEC 2.0 Descriptor Programmer's Guide
  - AN2932 Serial RapidIO® Bring-Up Procedure on PowerQUICC® III
  - AN2810 PowerQUICC UPM Configuration

Related Session Resources

Sessions

Session ID	Title
AN336	MPC8568 Primer
AN355	Technical Overview of the MPC8568 PowerQUICC™ III Processor for Integrated Networking and Control
AN359	Performance Optimization of QUICC Engine™ Architecture-Based Systems - Tips, Tricks and Trade-Offs

Demos

Pedestal ID	Demo Title
503	Programmable Network Interworking on the QUICC Engine
502	IP Forwarding Using the MPC8360 AMC
510	High Density Media Gateway Demo

Meet the FSL Experts

Title	Time	Location
QE Birds-of-a-Feather	12:00 PM	Tuscany A

Please complete the session survey on your nTAG before you leave.

freescale™
semiconductor