Lock handling Library

This library provides access to hardware and software locks for use in concurrent C programs. In general it is not safe to use these to marshall within XC due to the assumptions XC makes about safe concurrent data access.

Two types of locks are provided. Hardware locks are fast and power efficient but there are a limited number per tile. Software locks are slower but you can use an unlimited number of them.

Software version and dependencies

This document pertains to version 2.0.1 of this library. It is known to work on version 14.0.2 of the xTIMEcomposer tools suite, it may work on other versions.

The library does not have any dependencies (i.e. it does not rely on any other libraries).

1 Hardware lock API

Type	hwlock_t
Description	This type represents a hardware lock.

Function	hwlock_alloc
Description	Allocate a hardware lock. This function will allocate a new hardware lock from the pool of hardware locks available on the xCORE. The hardware has a limited number of hardware locks (for example, current L and S series devices have 4 locks per tile).
Type	`hwlock_t hwlock_alloc(void)`
Returns	the allocated lock if allocation is successful or the value HWLOCK_NOT_ALLOCATED if not.

Function	hwlock_free
Description	Free a hardware lock. This function frees a given hardware lock and returns it to the hardware pool to be reallocated elsewhere.
Type	`void hwlock_free(hwlock_t lock)`
Parameters	`lock` the hardware lock to be freed. If this is an invalid lock id or not an currently allocated lock then the function will trap.

Function	hwlock_acquire
Description	Acquire a hardware lock. This function acquires a lock for the current logical core. If another core holds the lock the function will pause until the lock is released.
Type	`void hwlock_acquire(hwlock_t lock)`
Parameters	`lock` the hardware lock to acquire

Function	hwlock_release
Description	Release a hardware lock. This function releases a lock from the current logical core. The lock should have been previously claimed by hwlock_acquire().
Type	`void hwlock_release(hwlock_t lock)`
Parameters	`lock` the hardware lock to release

2 Software lock API

Type	swlock_t
Description	Type that represents a software lock.

Macro	SWLOCK_INITIAL_VALUE
Description	This define should be used to initialize a software lock e.g. `swlock_t my_lock = SWLOCK_INITIAL_VALUE;` If you intialize this way there is no need to call swlock_init().

Function	swlock_init
Description	Initialize a software lock. This function will initialize a software lock for use. Note that unlike hardware locks, there is no need to allocate or free a software lock from a limited pool.
Type	`void swlock_init(swlock_t &lock)`

Function	swlock_try_acquire
Description	Try and acquire a software lock. This function tries to acquire a lock for the current logical core. If another core holds the lock then the function will fail and return.
Type	`int swlock_try_acquire(swlock_t &lock)`
Parameters	`lock` the software lock to acquire.
Returns	a value that is equal to SWLOCK_NOT_ACQUIRED if the attempt fails. Any other value indicates that the acquisition has succeeded.

Function	swlock_acquire
Description	Acquire a software lock. This function acquires a lock for the current logical core. If another core holds the lock then the function will wait until it becomes available.
Type	`void swlock_acquire(swlock_t &lock)`
Parameters	`lock` the software lock to acquire.

Function	swlock_release
Description	Release a software lock. This function releases a previously acquired software lock for other cores to use.
Type	`void swlock_release(swlock_t &lock)`
Parameters	`lock` the software lock to release.

APPENDIX A - Known Issues

No known issues.

APPENDIX B - Locks library change log

B.1 2.0.1

- Update to use lock resource macro from the standard library to prevent compile warning

B.2 2.0.0

- Restructured library