What and why single-cycle routers?

Single-cycle routers are routers with router latency of 1. When there is no congestion, the packet can leave the router 1 cycle after it enters the router.

Single-cycle router is assumed when we want to minimize the effect that hardware constraints have on the network performance.

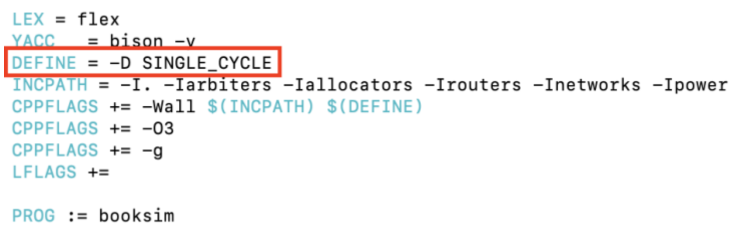
How to run single-cycle routers in BookSim?

Unfortunately, we are not able to configure the routers to be single-cycle without modifying the code. By default, the router latency in BookSim is 4 cycles. Changing the delays (`routing_delay`, `vc_alloc_delay`, `sw_alloc_delay`, `st_final_delay`) to 0 will cause an assertion error.

The `InternalStep` function in `iq_router.cpp` has to be modified as follows.

```
void IQRouter::_InternalStep( )
#ifdef SINGLE_CYCLE
{
  if(!_active) {
    return;
  }

  _InputQueuing( );

  bool activity = !_proc_credits.empty();

  if(!_route_vcs.empty()){
    _RouteEvaluate( );
  }

  if(!_route_vcs.empty()) {
    _RouteUpdate( );
    activity = activity || !_route_vcs.empty();
  }

  if(_vc_allocator) {
    _vc_allocator->Clear();
    if(!_vc_alloc_vcs.empty()){
      _VCAllocEvaluate( );
    }
  }

  if(!_vc_alloc_vcs.empty()) {
    _VCAllocUpdate( );
    activity = activity || !_vc_alloc_vcs.empty();
  }

  if(_hold_switch_for_packet) {
    if(!_sw_hold_vcs.empty()){
      _SWHoldEvaluate( );
    }
  }

  if(_hold_switch_for_packet) {
    if(!_sw_hold_vcs.empty()) {
      _SWHoldUpdate( );
      activity = activity || !_sw_hold_vcs.empty();
    }
  }

  _sw_allocator->Clear();
  if(_spec_sw_allocator)
    _spec_sw_allocator->Clear();


  if(!_sw_alloc_vcs.empty()){
    _SWAllocEvaluate( );
  }

  if(!_sw_alloc_vcs.empty()) {
    _SWAllocUpdate( );
    activity = activity || !_sw_alloc_vcs.empty();
  }

  if(!_crossbar_flits.empty()){
    _SwitchEvaluate( );
  }

  if(!_crossbar_flits.empty()) {
    _SwitchUpdate( );
    activity = activity || !_crossbar_flits.empty();
  }

  _active = activity;

  _OutputQueuing( );

  _bufferMonitor->cycle( );
  _switchMonitor->cycle( );
}

#else
{
  if(!_active) {
    return;
  }

  _InputQueuing( );
  bool activity = !_proc_credits.empty();

  if(!_route_vcs.empty())
    _RouteEvaluate( );
  if(_vc_allocator) {
    _vc_allocator->Clear();
    if(!_vc_alloc_vcs.empty())
      _VCAllocEvaluate( );
  }
  if(_hold_switch_for_packet) {
    if(!_sw_hold_vcs.empty())
      _SWHoldEvaluate( );
  }
  _sw_allocator->Clear();
  if(_spec_sw_allocator)
    _spec_sw_allocator->Clear();
  if(!_sw_alloc_vcs.empty())
    _SWAllocEvaluate( );
  if(!_crossbar_flits.empty())
    _SwitchEvaluate( );

  if(!_route_vcs.empty()) {
    _RouteUpdate( );
    activity = activity || !_route_vcs.empty();
  }
  if(!_vc_alloc_vcs.empty()) {
    _VCAllocUpdate( );
    activity = activity || !_vc_alloc_vcs.empty();
  }
  if(_hold_switch_for_packet) {
    if(!_sw_hold_vcs.empty()) {
      _SWHoldUpdate( );
      activity = activity || !_sw_hold_vcs.empty();
    }
  }
  if(!_sw_alloc_vcs.empty()) {
    _SWAllocUpdate( );
    activity = activity || !_sw_alloc_vcs.empty();
  }
  if(!_crossbar_flits.empty()) {
    _SwitchUpdate( );
    activity = activity || !_crossbar_flits.empty();
  }

  _active = activity;

  _OutputQueuing( );

  _bufferMonitor->cycle( );
  _switchMonitor->cycle( );
}
#endif
```

The above code has 2 versions of InternalStep: single-cycle and original BookSim code, separated by conditional macros. To use the single-cycle router code, we have to `#define SINGLE_CYCLE`, which can be done in the Makefile, as shown below.



```
LEX = flex
YACC   = bison -v
DEFINE = -D SINGLE_CYCLE
INCPATH = -I. -Iarbiters -Iallocators -Irouters -Inetworks -Ipower
CPPFLAGS += -Wall $(INCPATH) $(DEFINE)
CPPFLAGS += -O3
CPPFLAGS += -g
LFLAGS +=

PROG := booksim
```