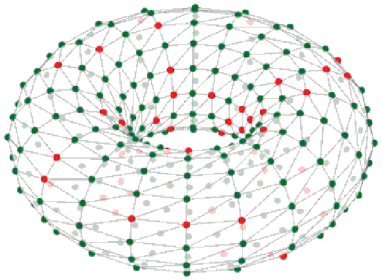
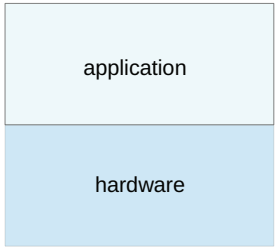
SpiNNaker API and event-driven simulation

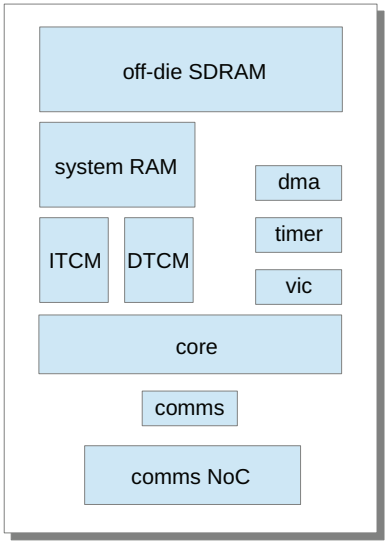


Luis Plana

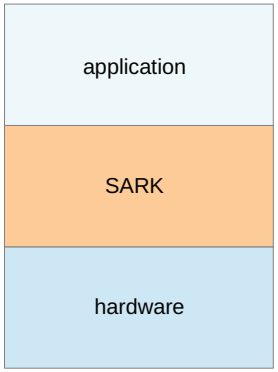
SpiNNaker Workshop, September 2016

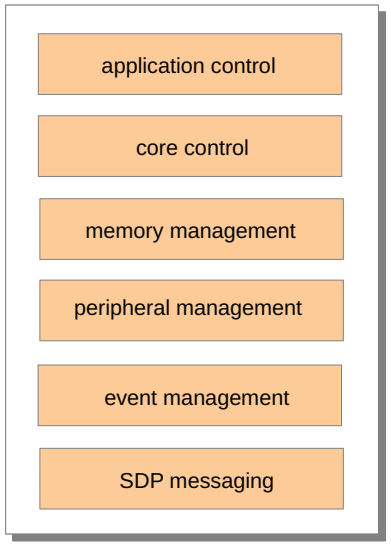
hardware resources



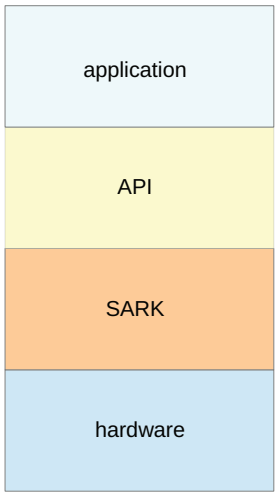


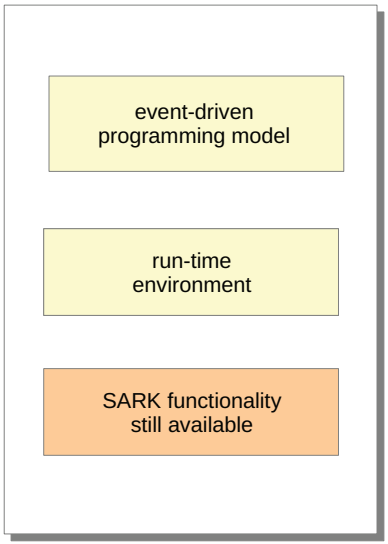
SARK: low-level software





API: run-time environment





event-driven model

applications do not control execution flow

applications indicate functions to be executed when events of interest occur

API controls execution and schedules application functions when appropriate

application functions are known as callbacks

events and callbacks

event	trigger
timer tick	periodic event has occurred
multicast packet received	multicast packet has arrived
DMA transfer done	scheduled DMA transfer completed successfully
SDP packet received	SDP packet has arrived
user event	application-triggered event has occurred

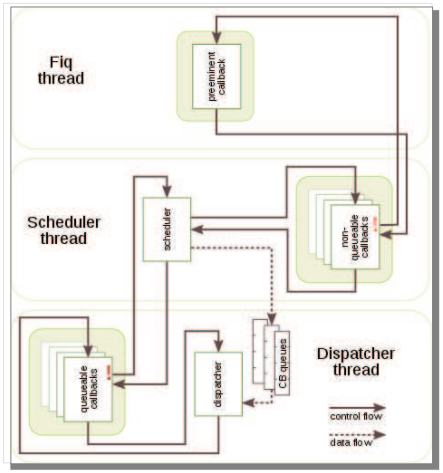
event	first argument	second argument
timer tick	simulation time (ticks)	null
MCP w/o payload received	key	0
MCP with payload received	key	payload
DMA transfer done	transfer ID	tag
SDP packet received	*mailbox	destination port
user event	arg0	arg1

priorities

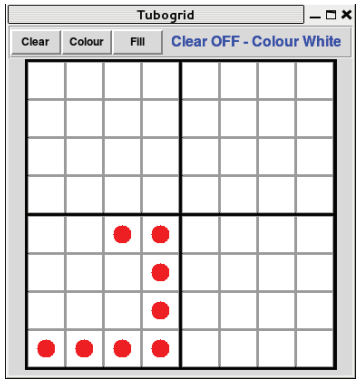
priority level = -1
only one callback
cannot be pre-empted

priority level = 0
can only be pre-empted
by priority -1 callback

priority level > 0
can be pre-empted
by priority <= 0 callbacks
scheduled in priority order



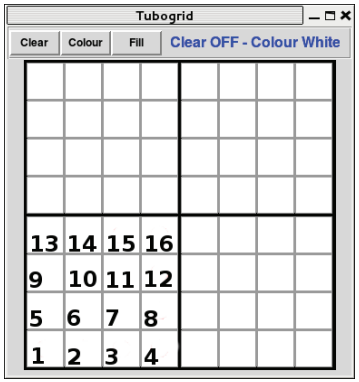
first program



first

```
// circle sequence
uint circle_pos[] =
{
  1, 2, 3, 4, 8, 12, 16, 15,
  14, 13, 9, 5, 6, 7, 11, 10
};

// iterate over 16 positions
for (uint i = 0; i < 16; i++)
{
  // update display,
  print_circle (circle_pos[i]);

  // and delay next circle
  for (uint j = 0; j < BIG_NUM; j++)
  {
    continue;
  }
}
```

distributed program



each core

```
// circle sequence
uint circle_pos[] =
{
  1, 2, 3, 4, 8, 12, 16, 15,
  14, 13, 9, 5, 6, 7, 11, 10
};

// this core's id
id = spin1_get_core_id();

// delay my circle,
for (uint j = 0; j < (id * BIG_NUM); j++)
{
  continue;
}

// and update display
print_circle (circle_pos[id]);
```

event-driven program

c_main

```
// 0.125s tick period (in microseconds)
#define TIMER_TICK_PERIOD  125000

void c_main()
{
  // initialize variables and state
  // ---------------------------------------
  id = spin1_get_core_id();
  my_state = OFF;
  old_state = my_state;

  // prepare for execution
  // ---------------------------------------
  // set timer tick value
  spin1_set_timer_tick (TIMER_TICK_PERIOD);

  // register callbacks
  spin1_callback_on (
    MC_PACKET_RECEIVED, packet, -1);

  spin1_callback_on (
    TIMER_TICK, timer, 0);

  // go
  // ---------------------------------------
  spin1_start(SYNC_WAIT);
}
```

packet callback

```
void packet (uint pkt_key, uint pkt_payload)
{
  // update my state
  my_state = ON;
}
```

timer callback

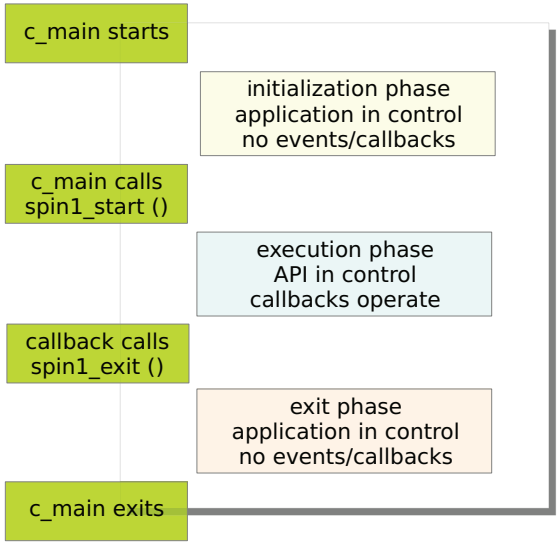
```
void timer (uint ticks, uint b)
{
  // check if state changed
  if (my_state != old_state)
  {
    // update display,
    print_circle (circle_pos[id]);

    // send a packet to next core in the chain,
    spin1_send_mc_packet(my_key, 0, NO_PAYLOAD);

    // and remember state
    old_state = my_state;
  }
}
```

additional support

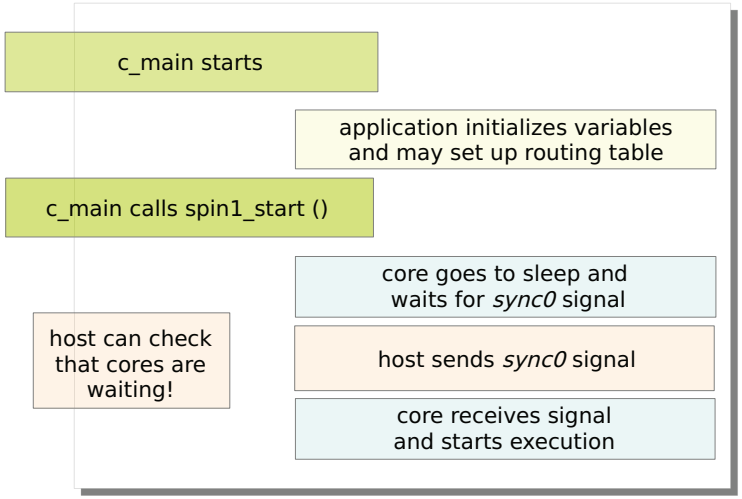
function	use
start/stop execution	start and stop simulation
set timer period	real-time or periodic callback
send multicast packet	inter-core communications
send SDP packet	host or I/O peripheral communications
start DMA transfer	software-managed cache
trigger user event	start a callback with priority <= 0
schedule callback	start a callback with priority > 0
enable/disable interrupts	critical section access (inter-thread control)
provide chip address and core ID	find out who you are
configure multicast routing table	setup routing entries

see API documentation for complete list

program structure



synchronization barrier



to think about: pitfalls

asynchronous operation and communications

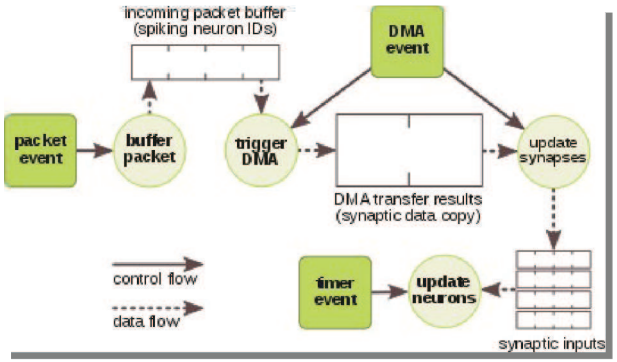
multicast packets can be dropped due to congestion

UDP-based I/O *not guaranteed!*

no floating-point support use fixed-point arithmetic

no globally-shared resources use message passing

example: spiking neural network



what is a sensible choice of priorities?