

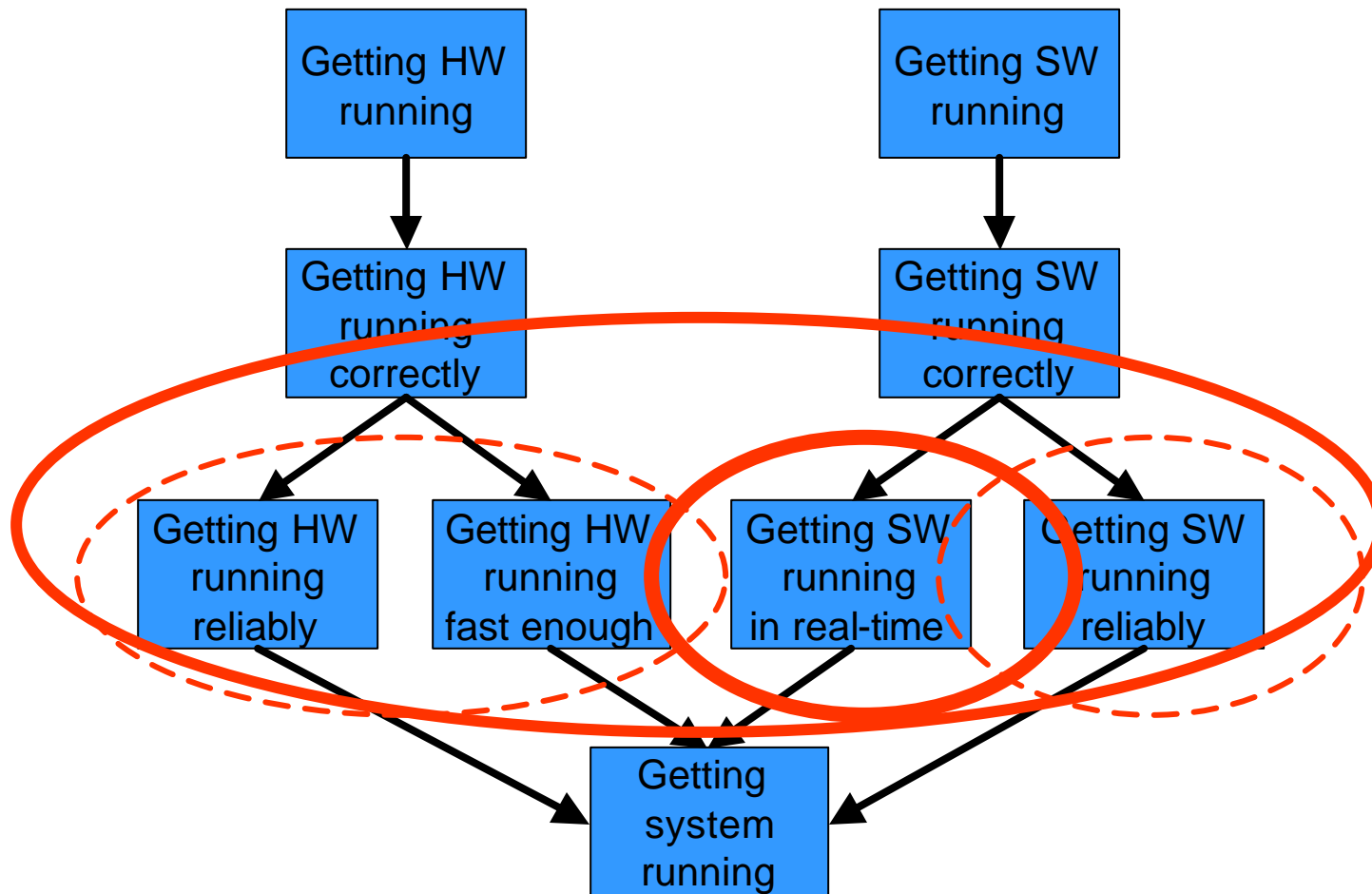
# Architecture, Design and Implementation of Embedded Systems for Real-Time Applications

CpE-450 - Spring 05  
Class 23

Bruce McNair  
bmcnair@stevens.edu

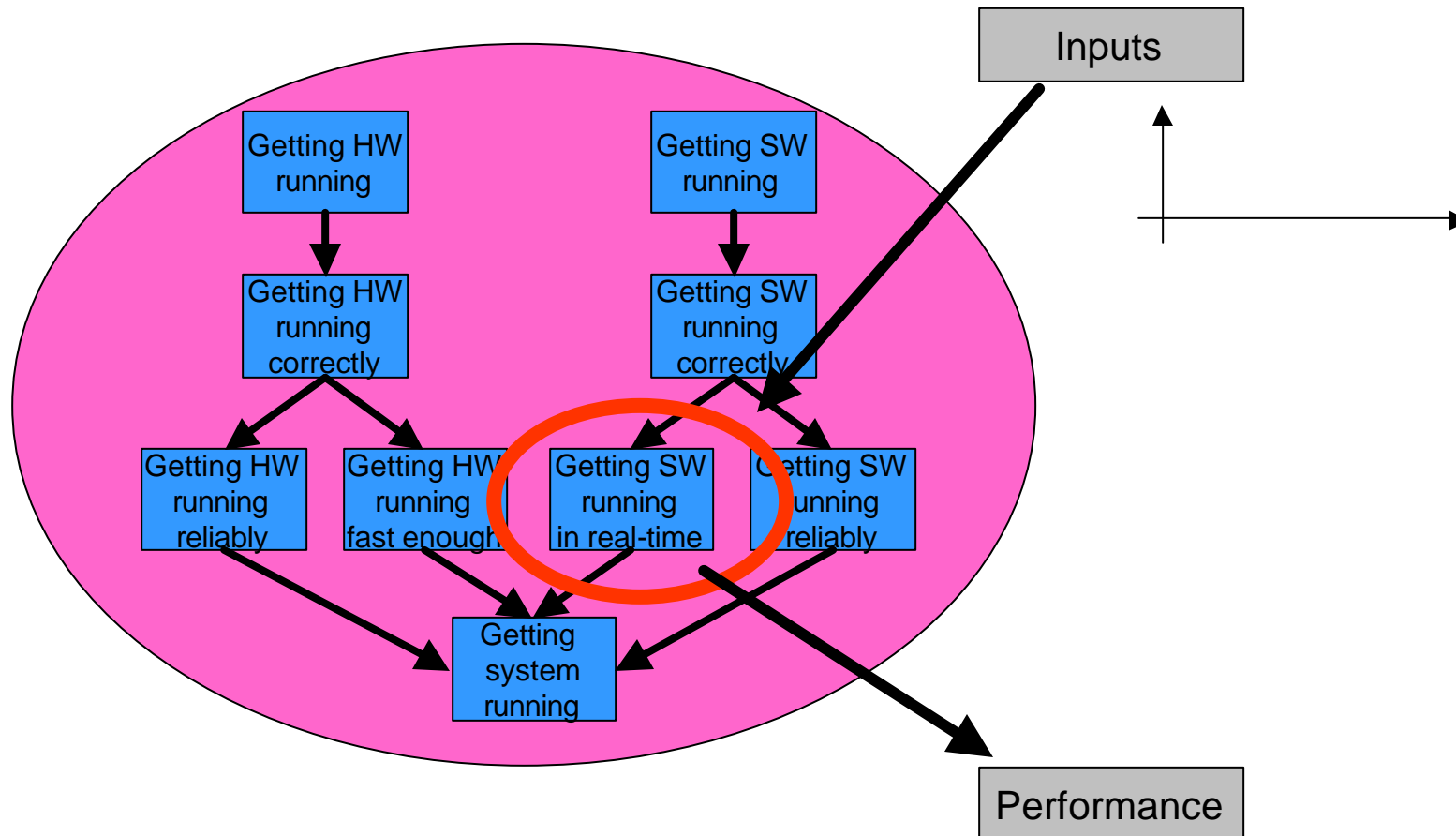
# Real-time Performance

- And other items



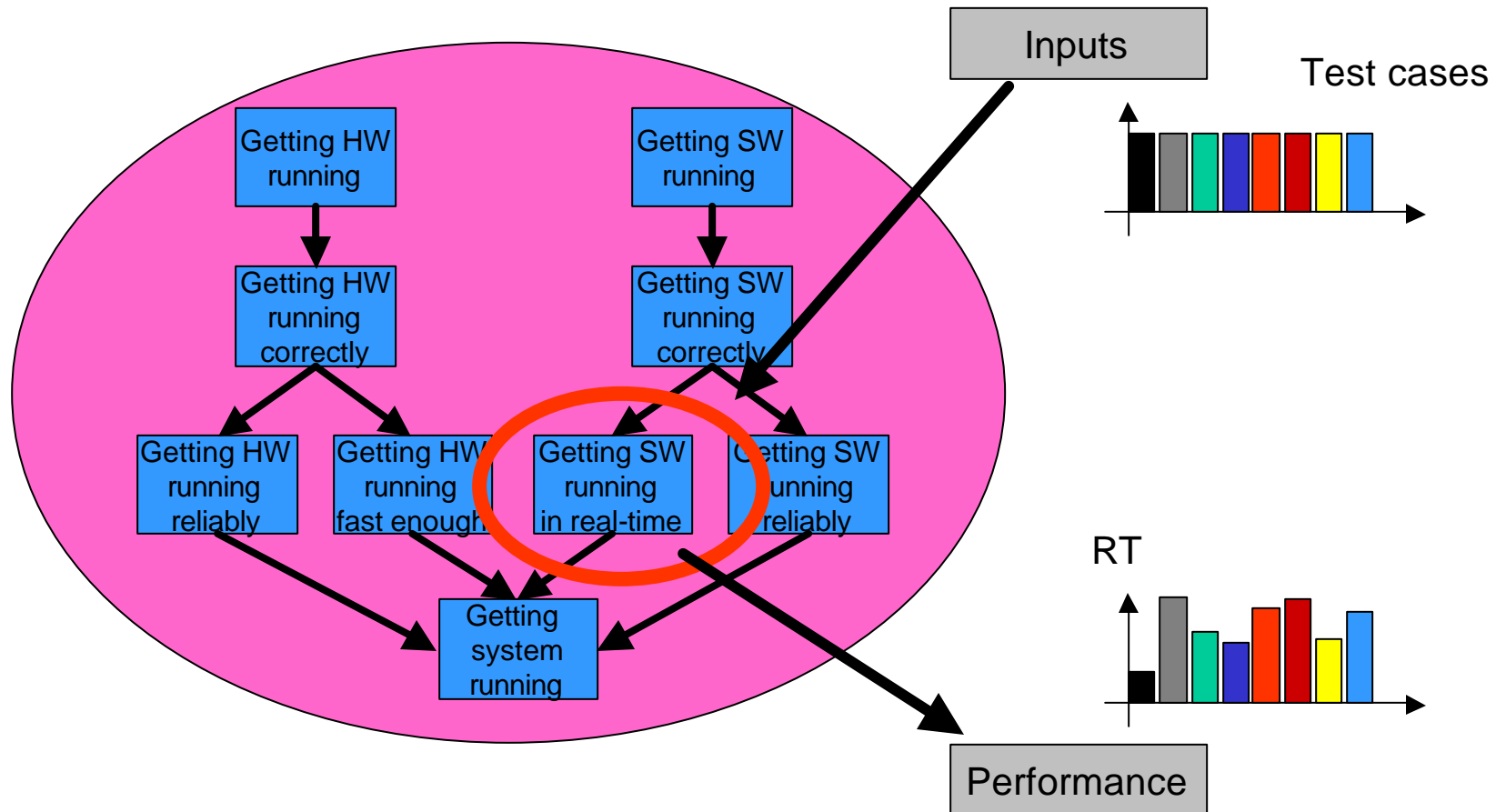
# Real-Time Performance

- Evaluating performance



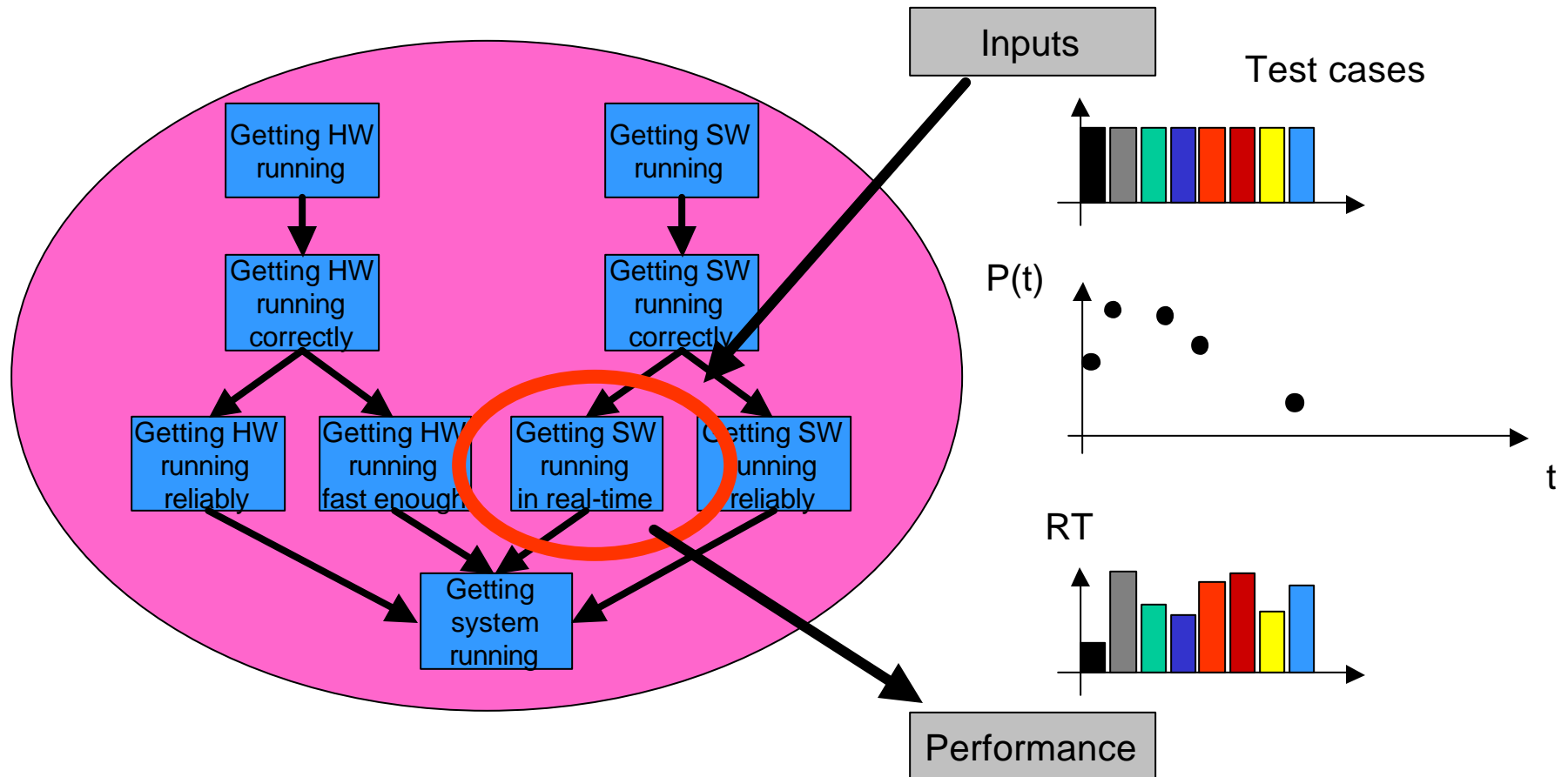
# Real-Time Performance

- Evaluating performance



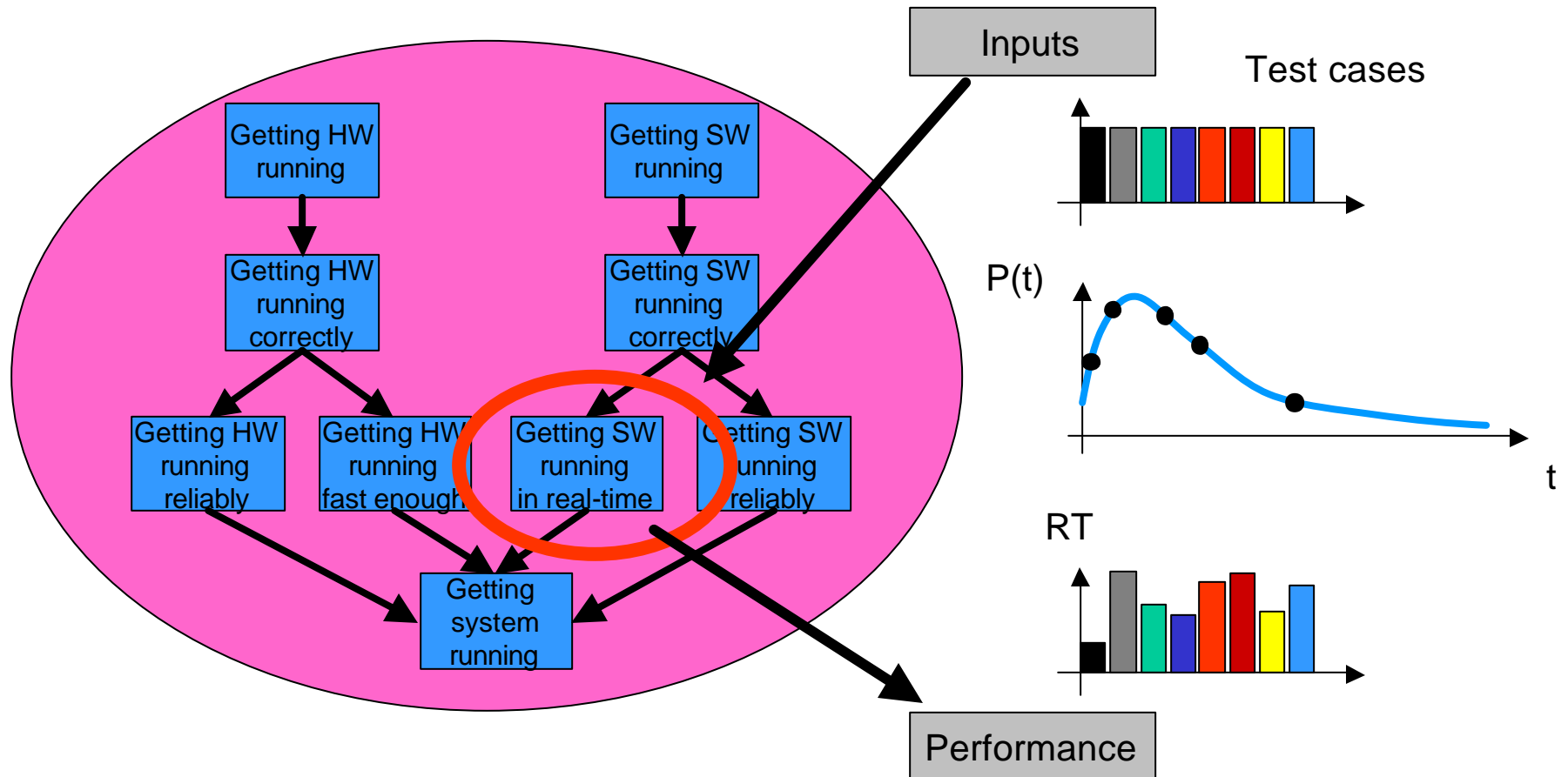
# Real-Time Performance

- Evaluating performance – a dynamically changing parameter



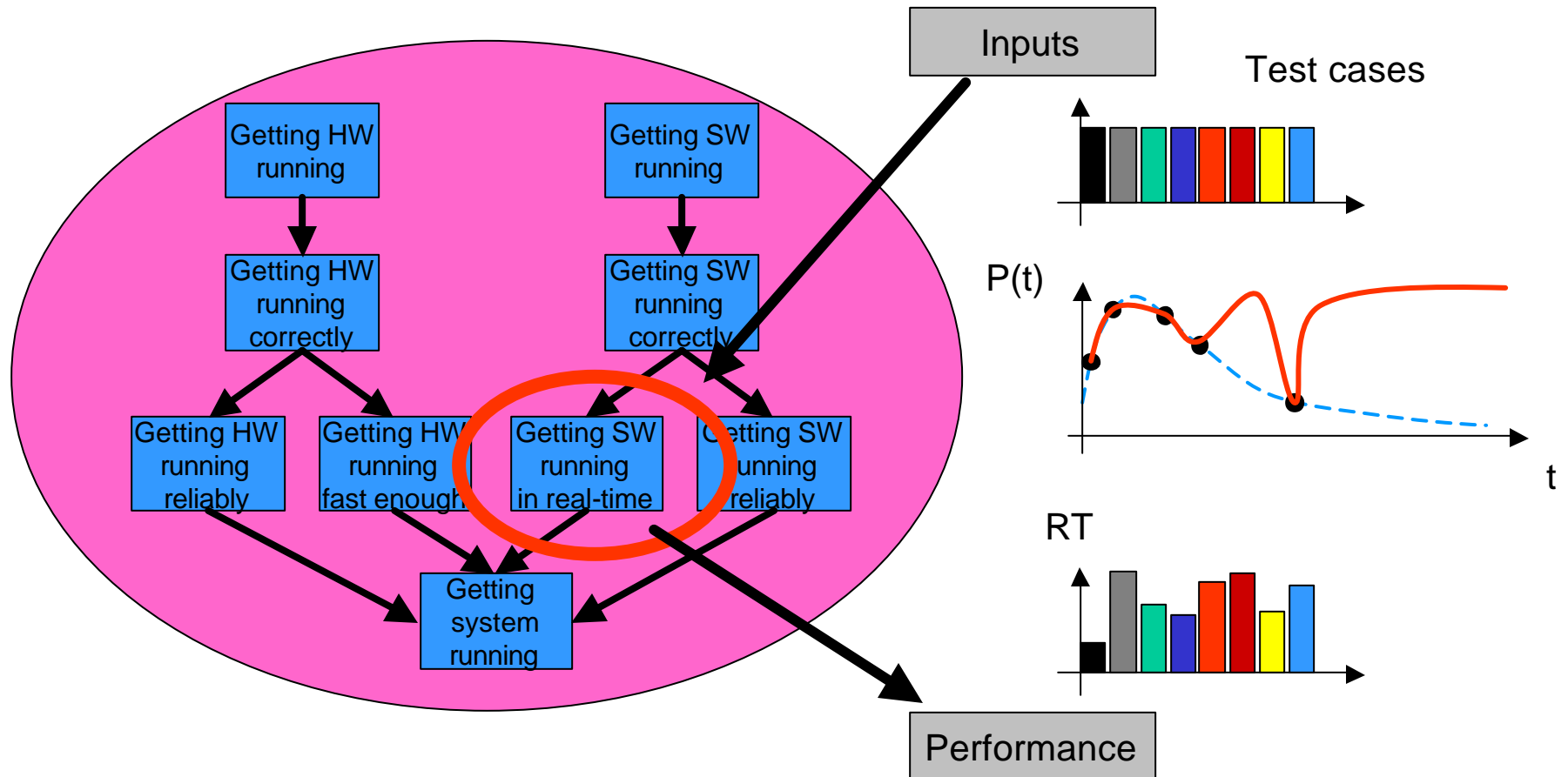
# Real-Time Performance

- Evaluating performance – a dynamically changing parameter

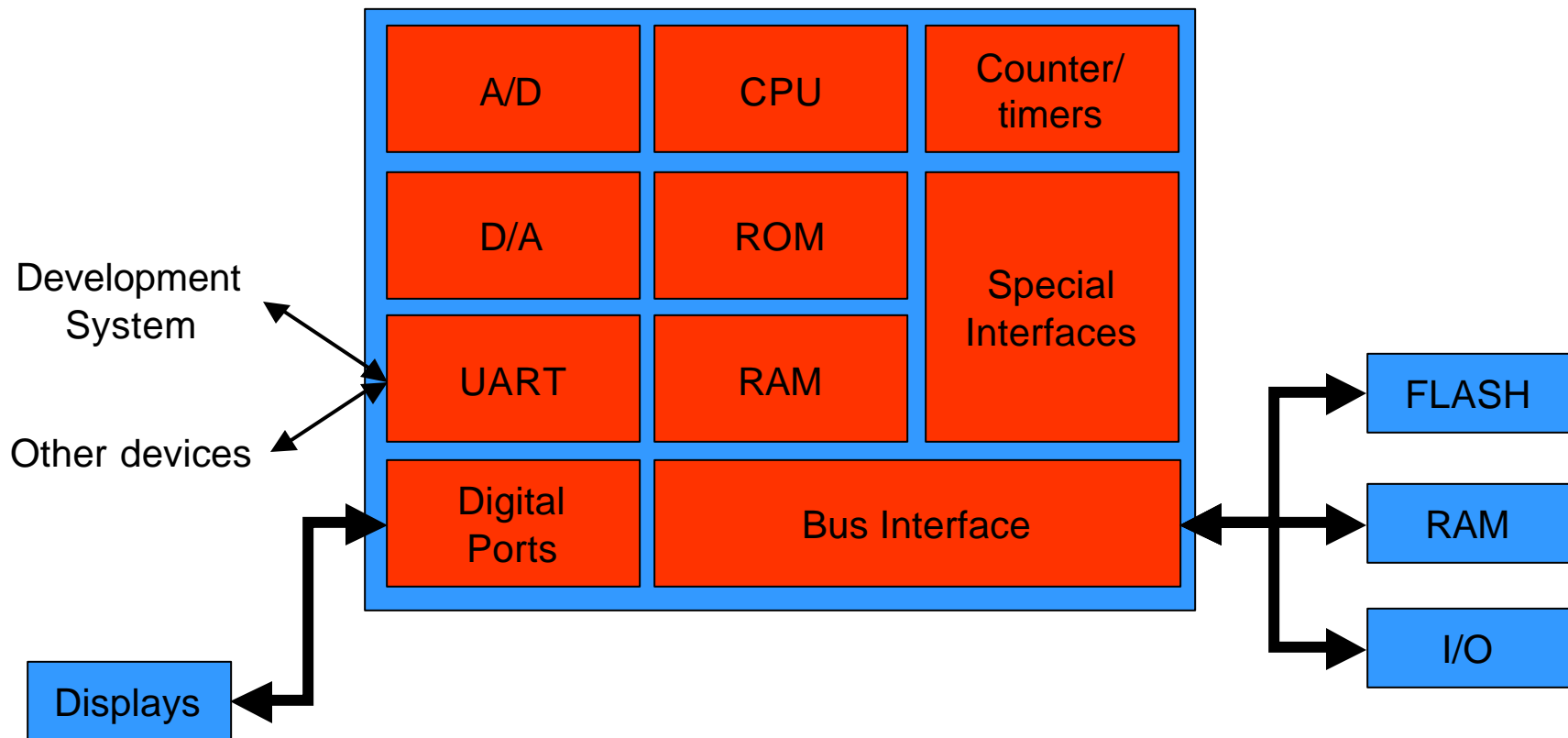


# Real-Time Performance

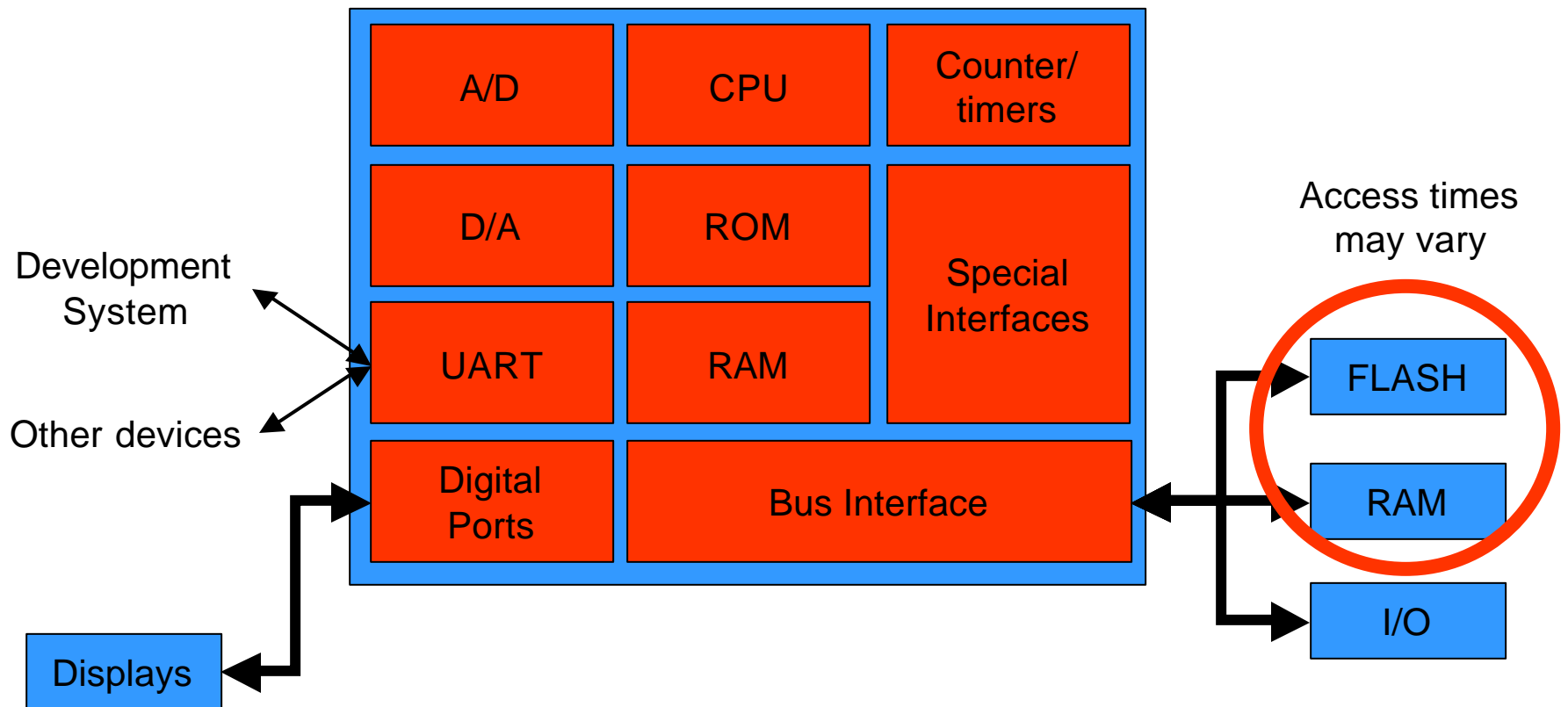
- Evaluating performance – a dynamically changing parameter



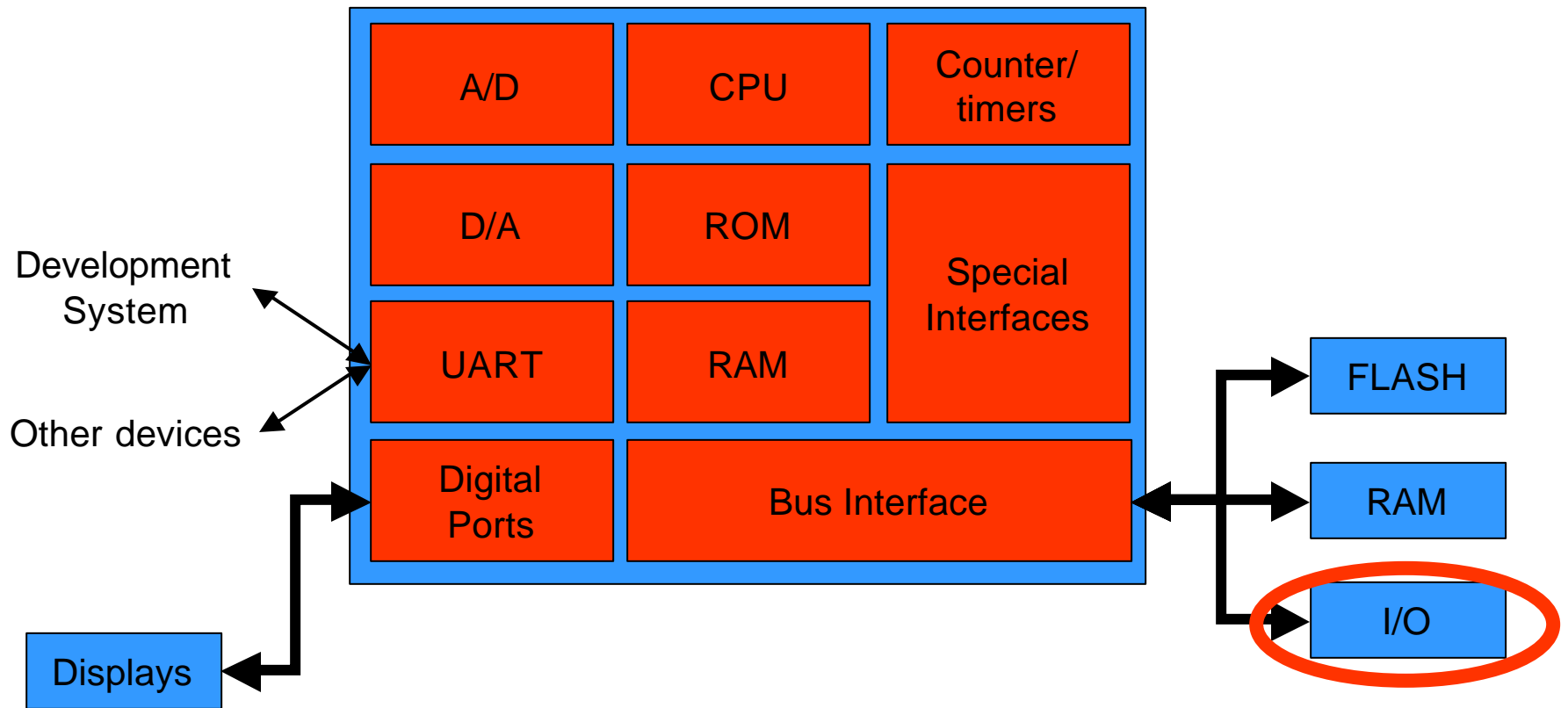
# A Representative Embedded System



# A Representative Embedded System



# A Representative Embedded System



Is I/O simulated for testing representative?

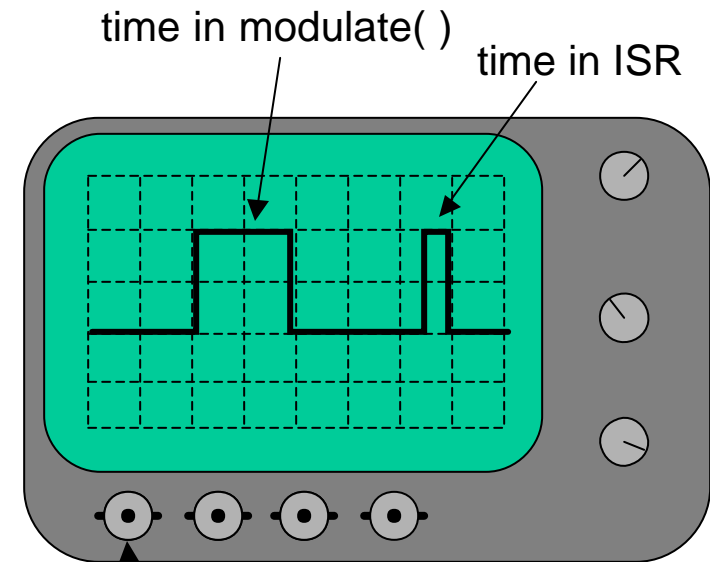
# Real-time Observability

- Observing time spent in ISR, other functions

```
float sine_table[SIZE];
float cosine_table[SIZE];
struct signal
{
    float real;
    float imag;
};

void main( )
{
    boolean bits[BLOCK_LENGTH];
    signal sig[BLOCK_LENGTH];
    float mod[BLOCK_LENGTH];
    initialize_sine_table(*sine_table, *cosine_table);
    set_up_interrupts(INTERRUPT_ON_DATA_PRESENT);
    while(1)
    {
        get_data(bits);
        generate_baseband(bits, sig);
        set_output_flag(1);
        modulate(sig, mod, BLOCK_LENGTH, Fc, delta_t);
        set_output_flag(0);
        output_mod(mod);
        sleep( );
    }
}

void interrupt_handler_DATA_PRESENT(void)
{
    set_output_flag(1);
    /* wake up main( ) when data arrives */
    set_output_flag(0);
}
```



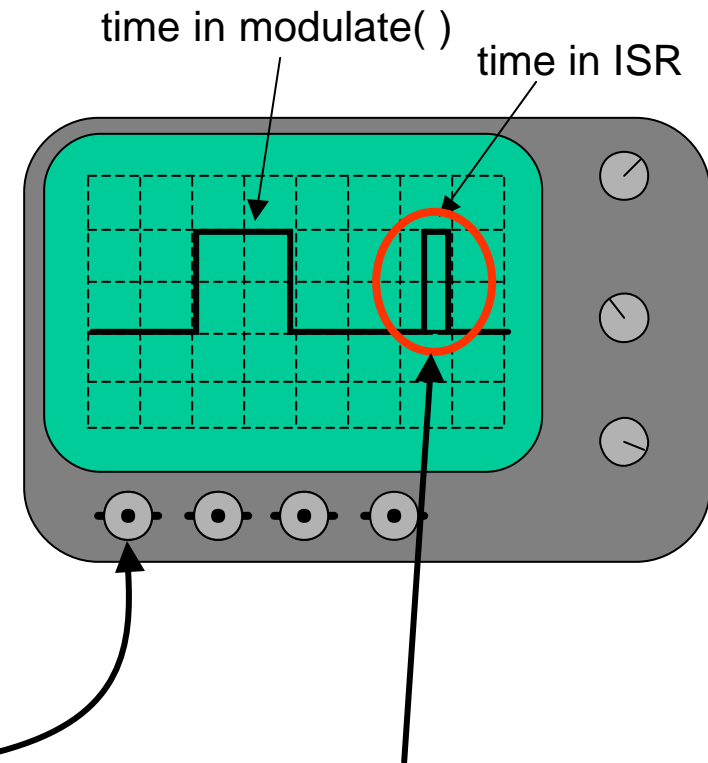
# Real-time Observability

- Observing time spent in ISR, other functions

```
float sine_table[SIZE];
float cosine_table[SIZE];
struct signal
{
    float real;
    float imag;
};

void main( )
{
    boolean bits[BLOCK_LENGTH];
    signal sig[BLOCK_LENGTH];
    float mod[BLOCK_LENGTH];
    initialize_sine_table(*sine_table, *cosine_table);
    set_up_interrupts(INTERRUPT_ON_DATA_PRESENT);
    while(1)
    {
        get_data(bits);
        generate_baseband(bits, sig);
        set_output_flag(1);
        modulate(sig, mod, BLOCK_LENGTH, Fc, delta_t);
        set_output_flag(0);
        output_mod(mod);
        sleep( );
    }
}

void interrupt_handler_DATA_PRESENT(void)
{
    set_output_flag(1);
    /* wake up main( ) when data arrives */
    set_output_flag(0);
}
```

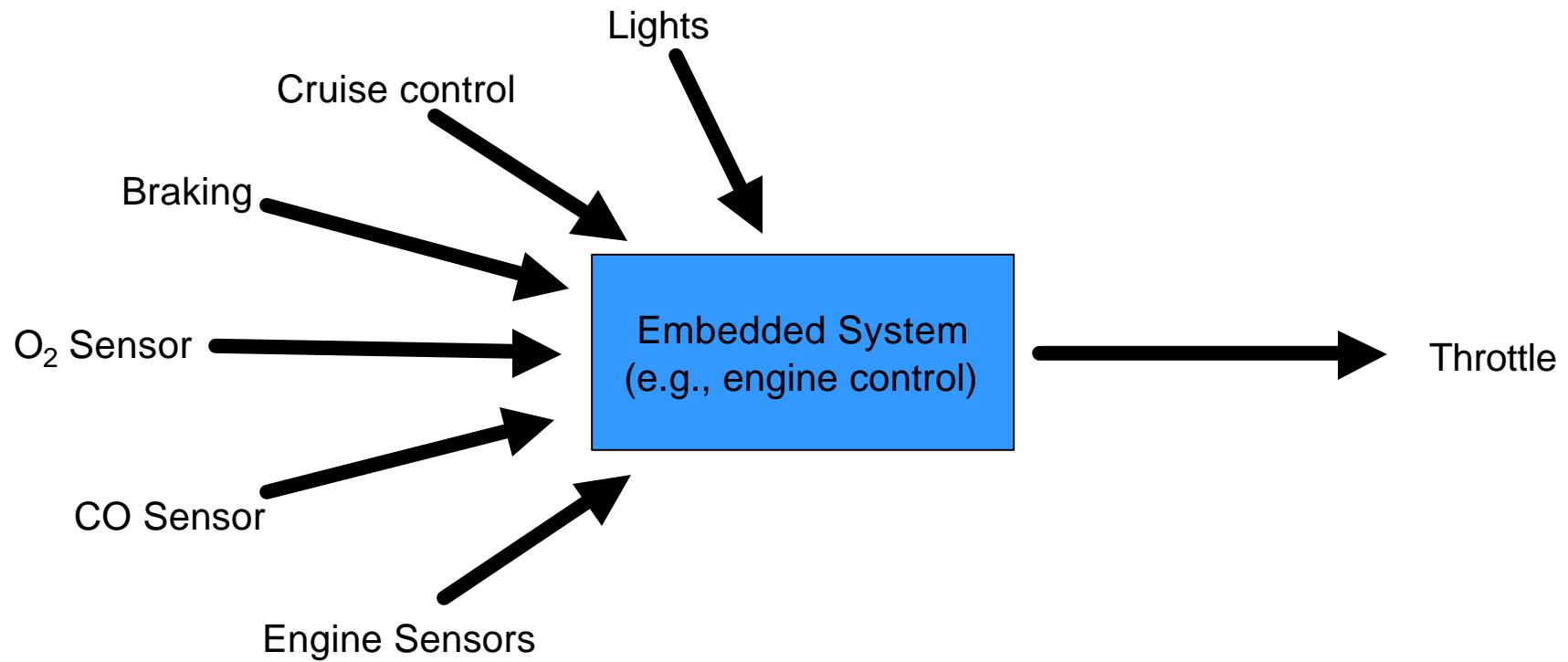


What if ISR is not called frequently?

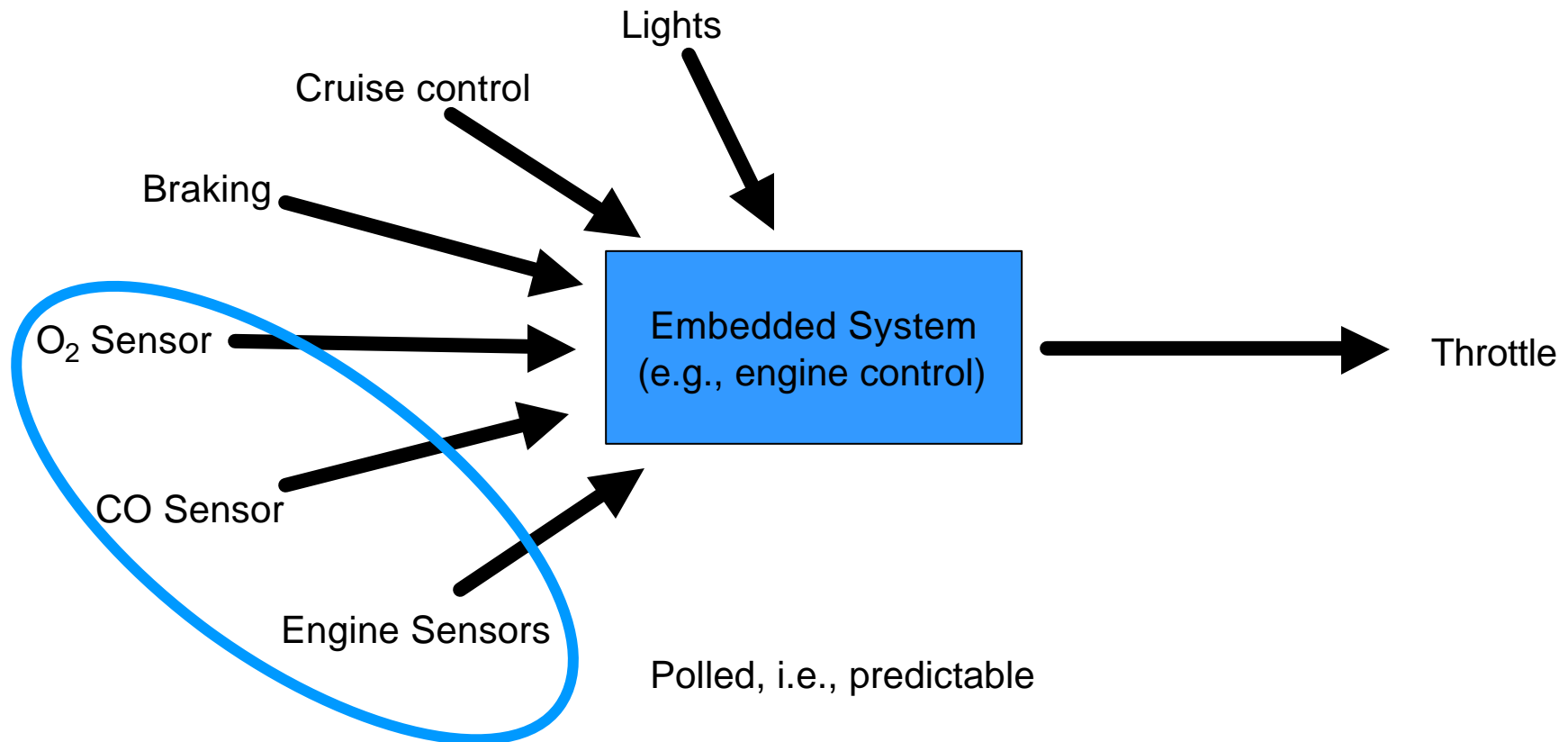
# Unpredictable data

- Diagram of embedded system:
  - External sensors
  - Interaction
  - Auto sudden acceleration problem

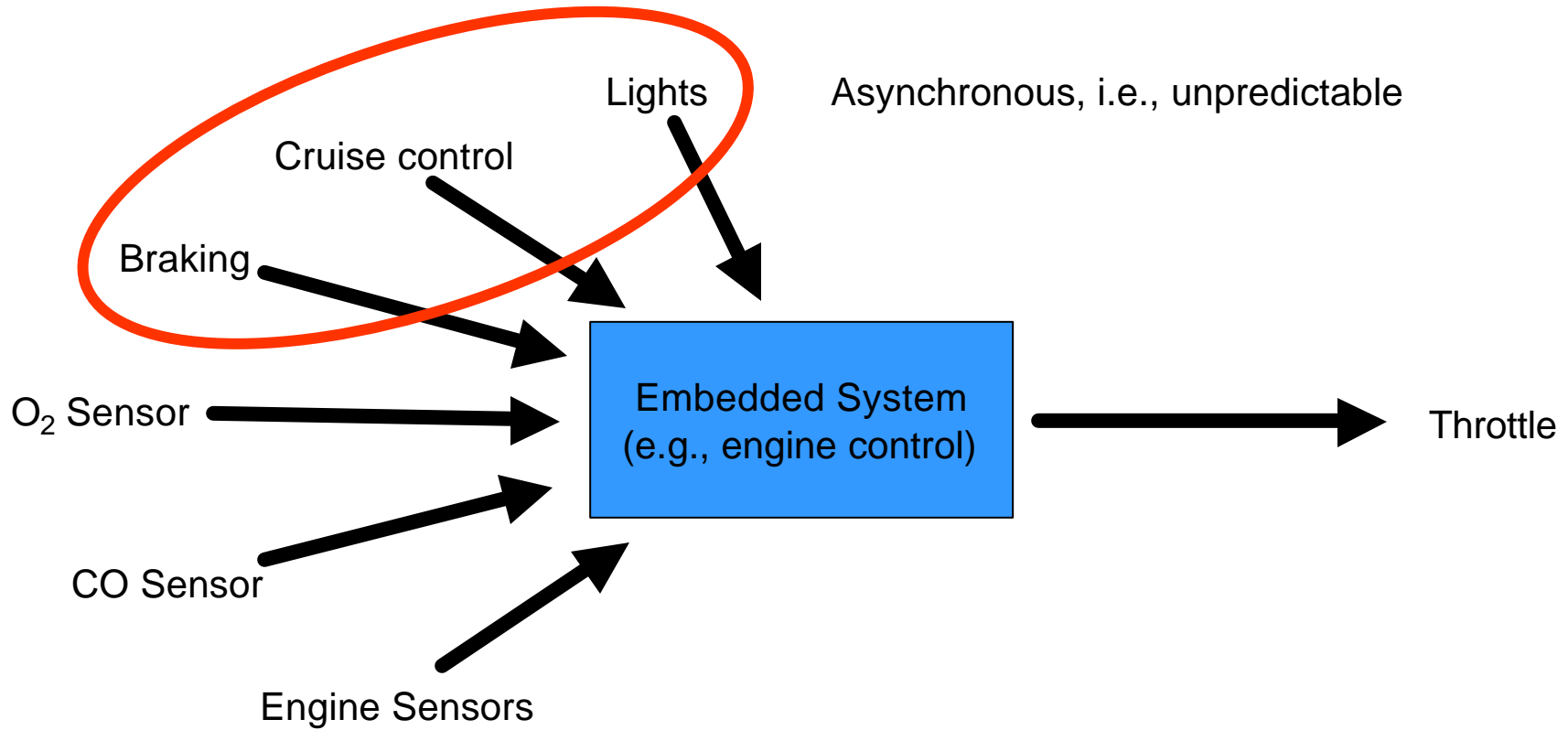
# Unpredictability of data, events



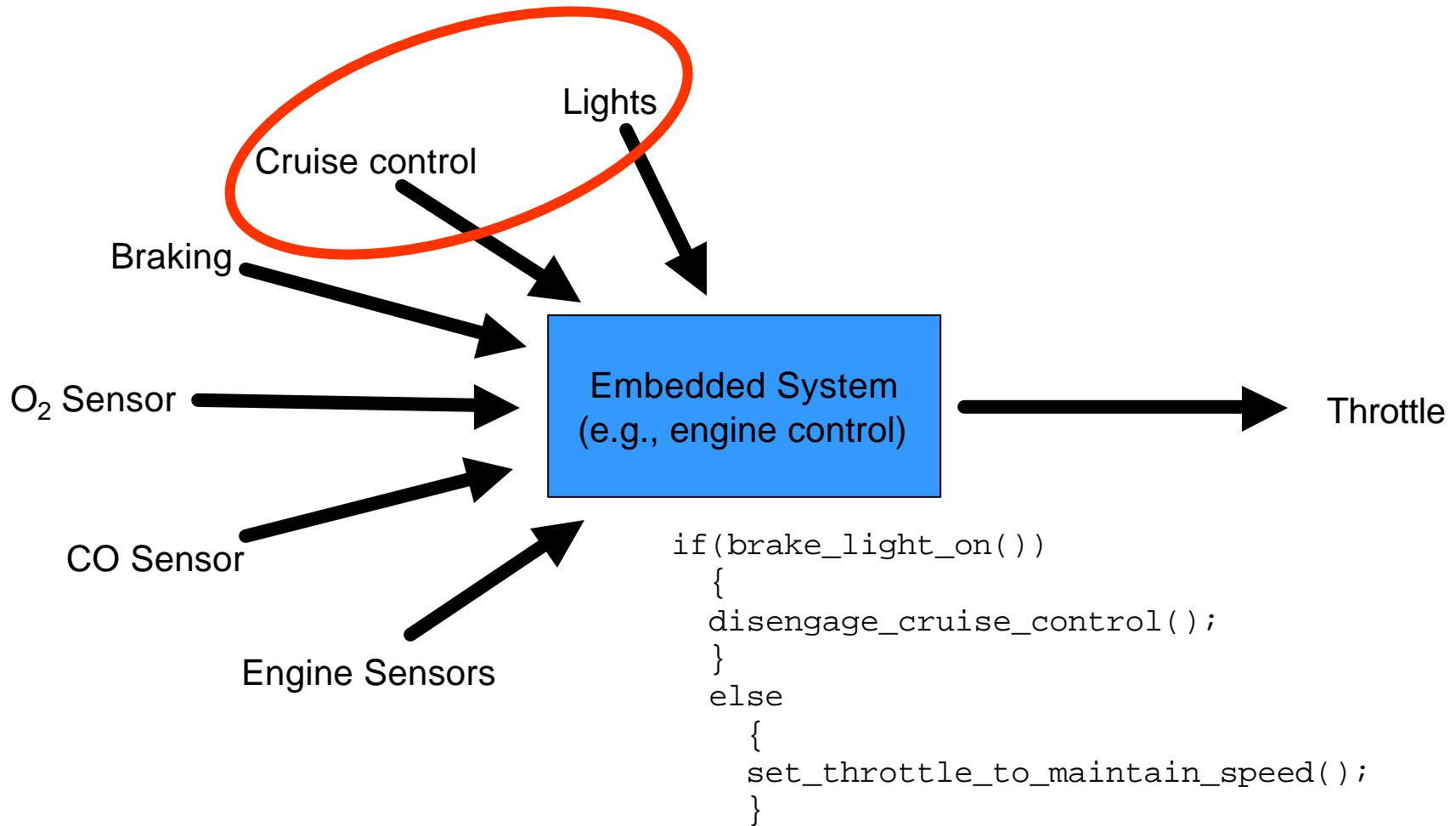
# Unpredictability of data, events



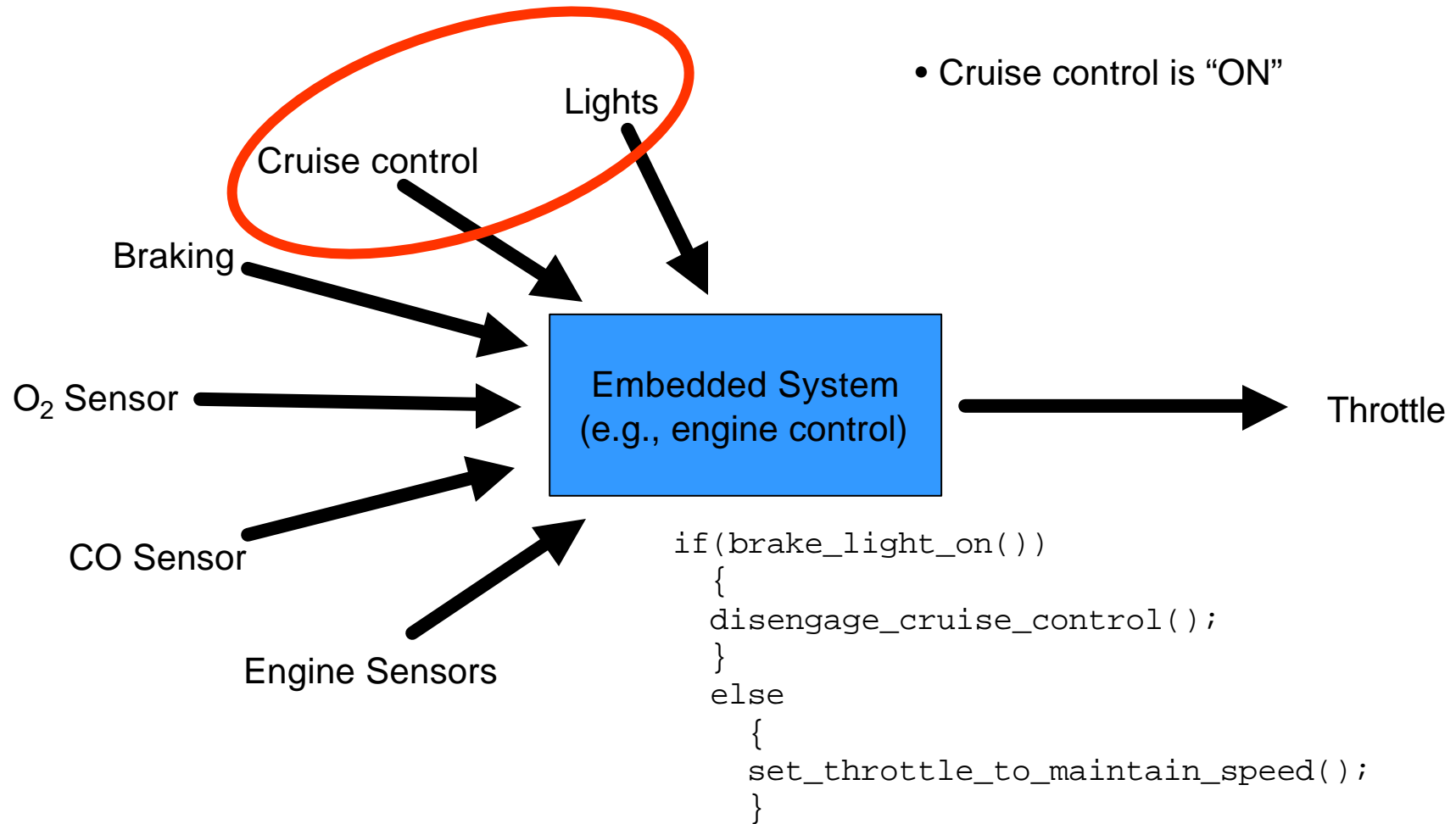
# Unpredictability of data, events



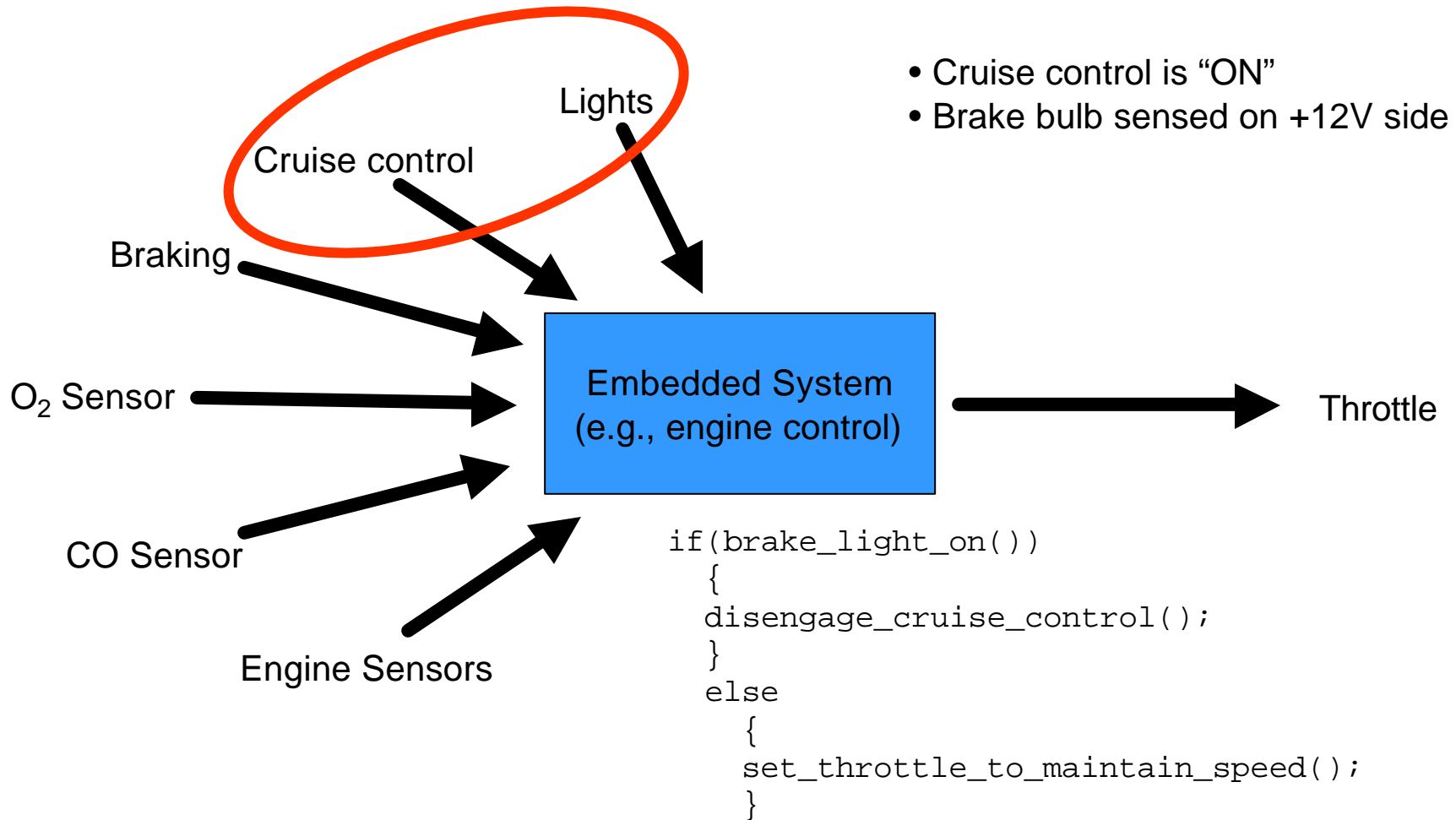
# Unanticipated interaction of data, events



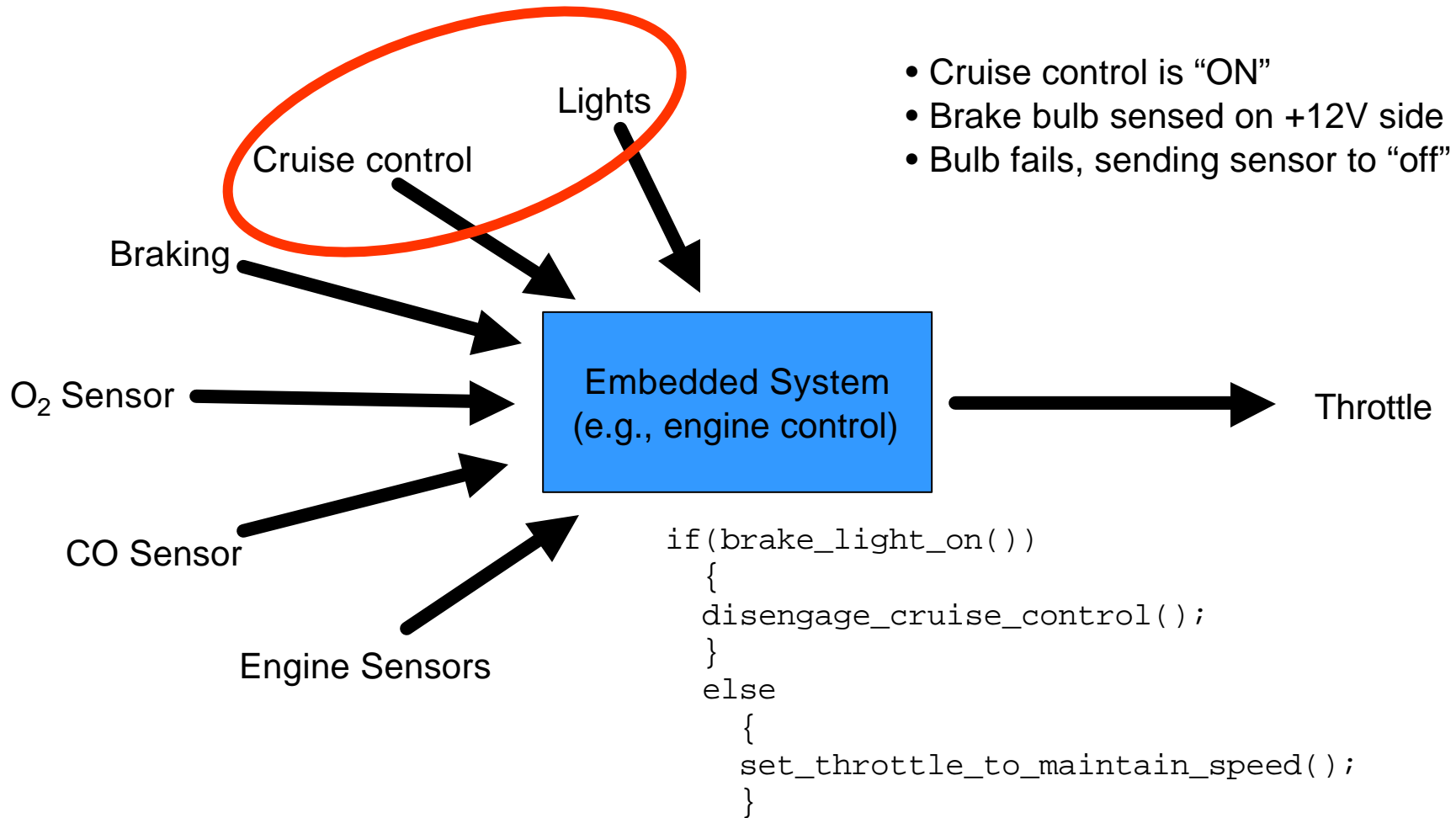
# Unanticipated interaction of data, events



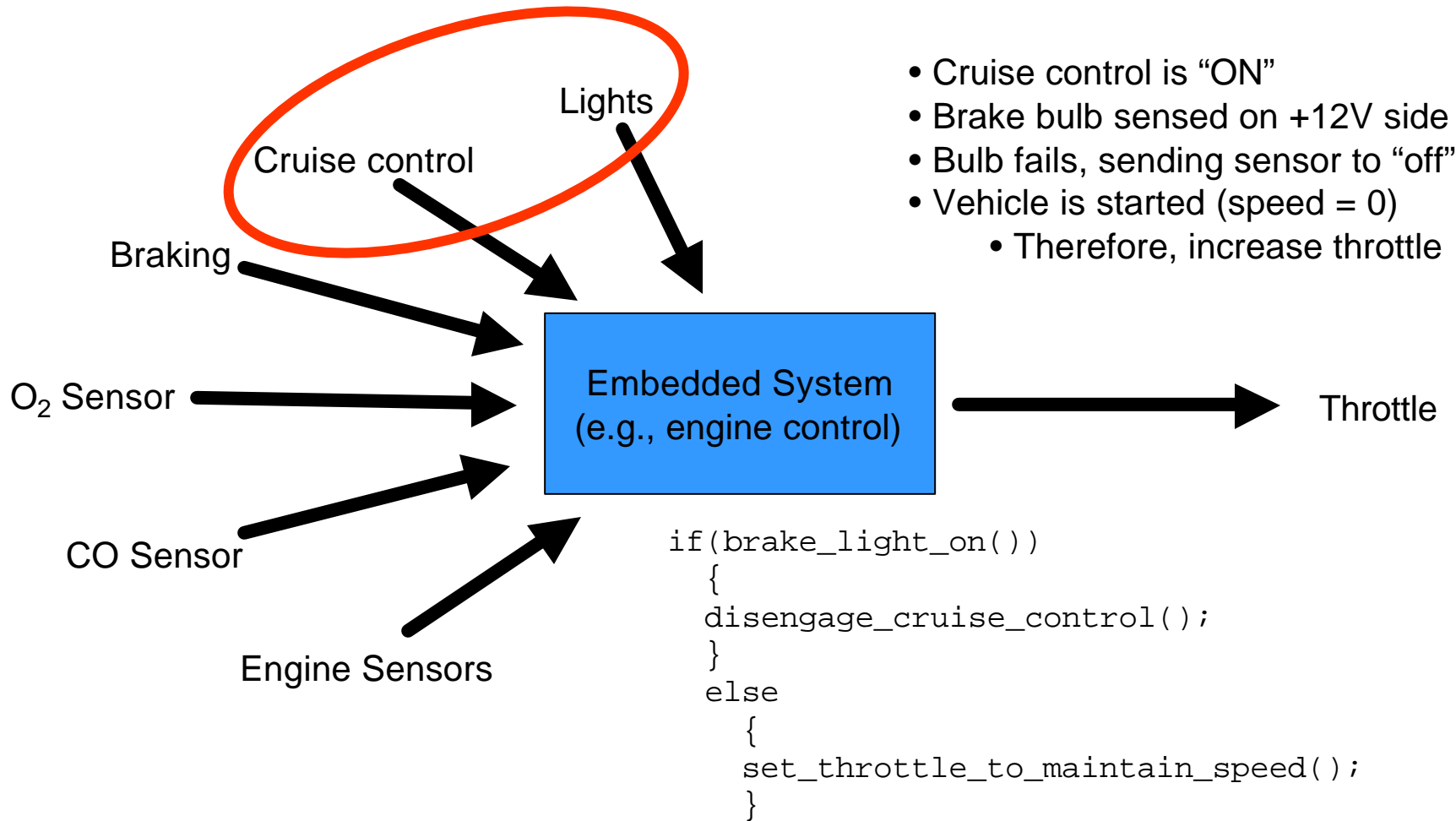
# Unanticipated interaction of data, events



# Unanticipated interaction of data, events



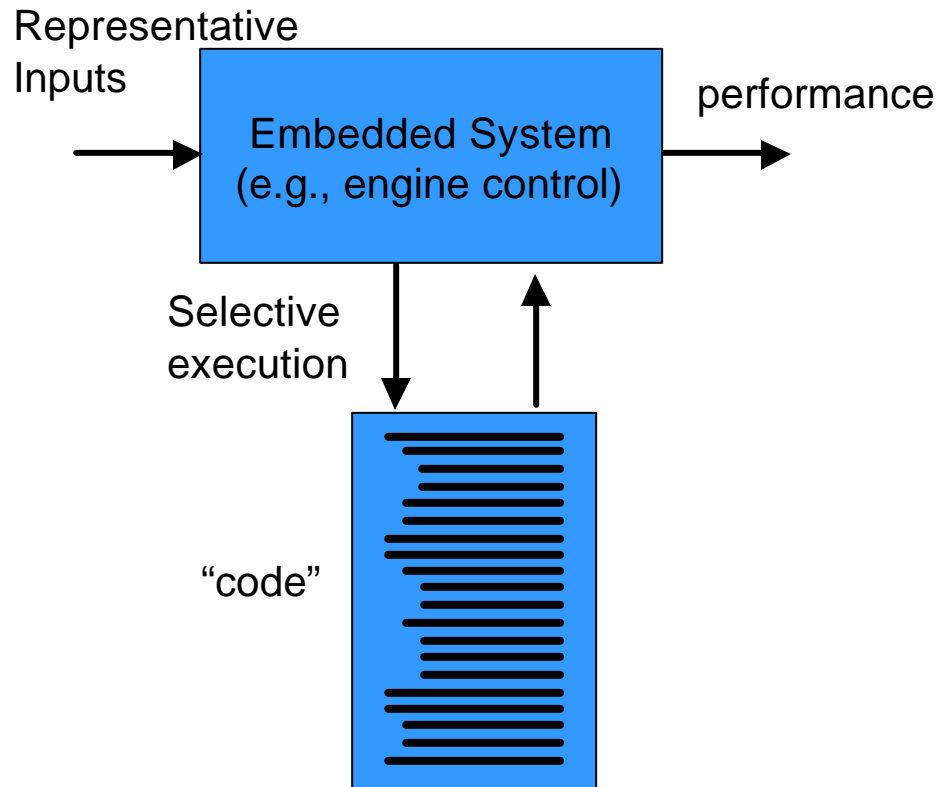
# Unanticipated interaction of data, events



- Cruise control is “ON”
- Brake bulb sensed on +12V side
- Bulb fails, sending sensor to “off”
- Vehicle is started (speed = 0)
  - Therefore, increase throttle

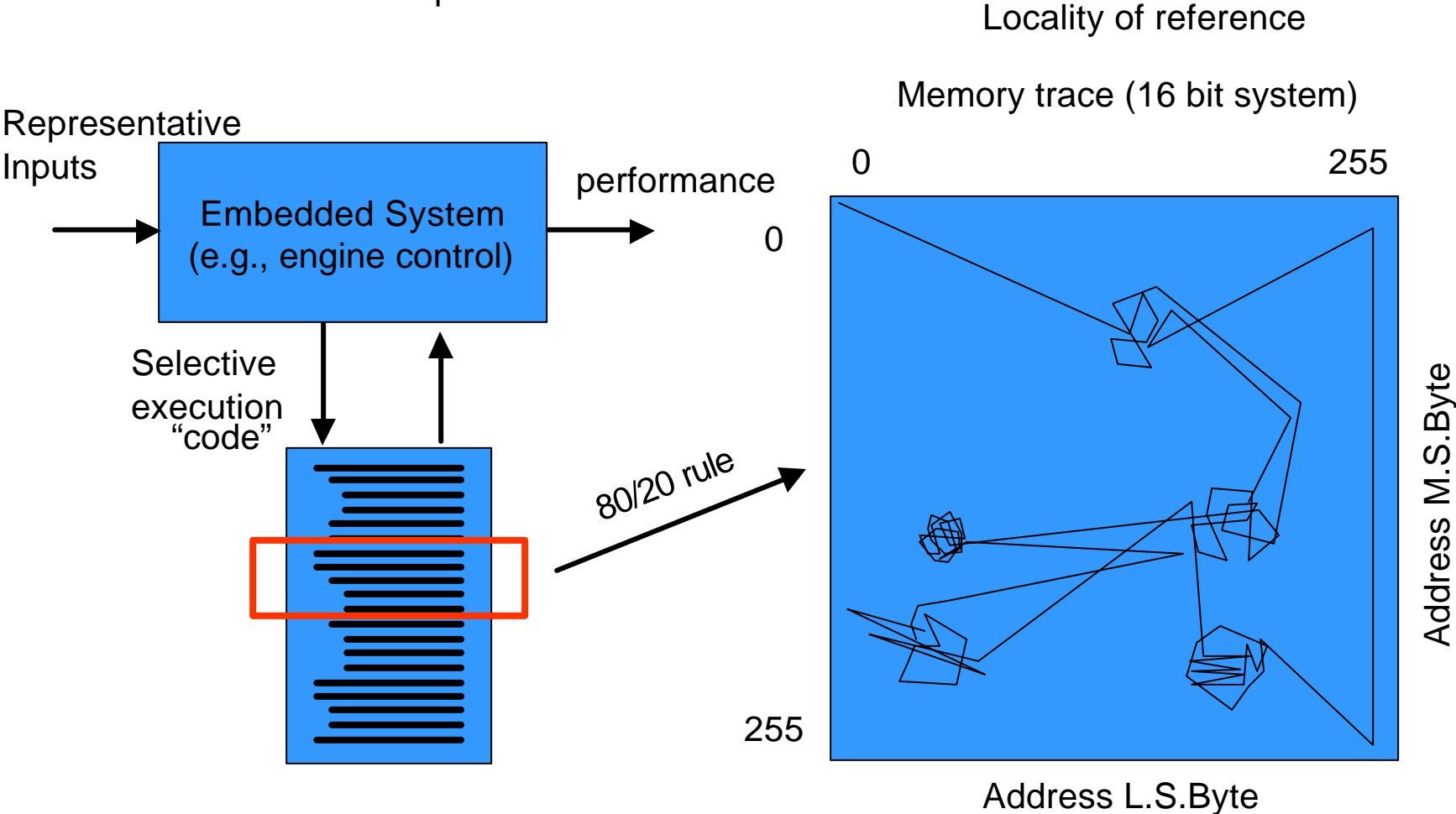
# Data sensitivity

# Benchmarking performance

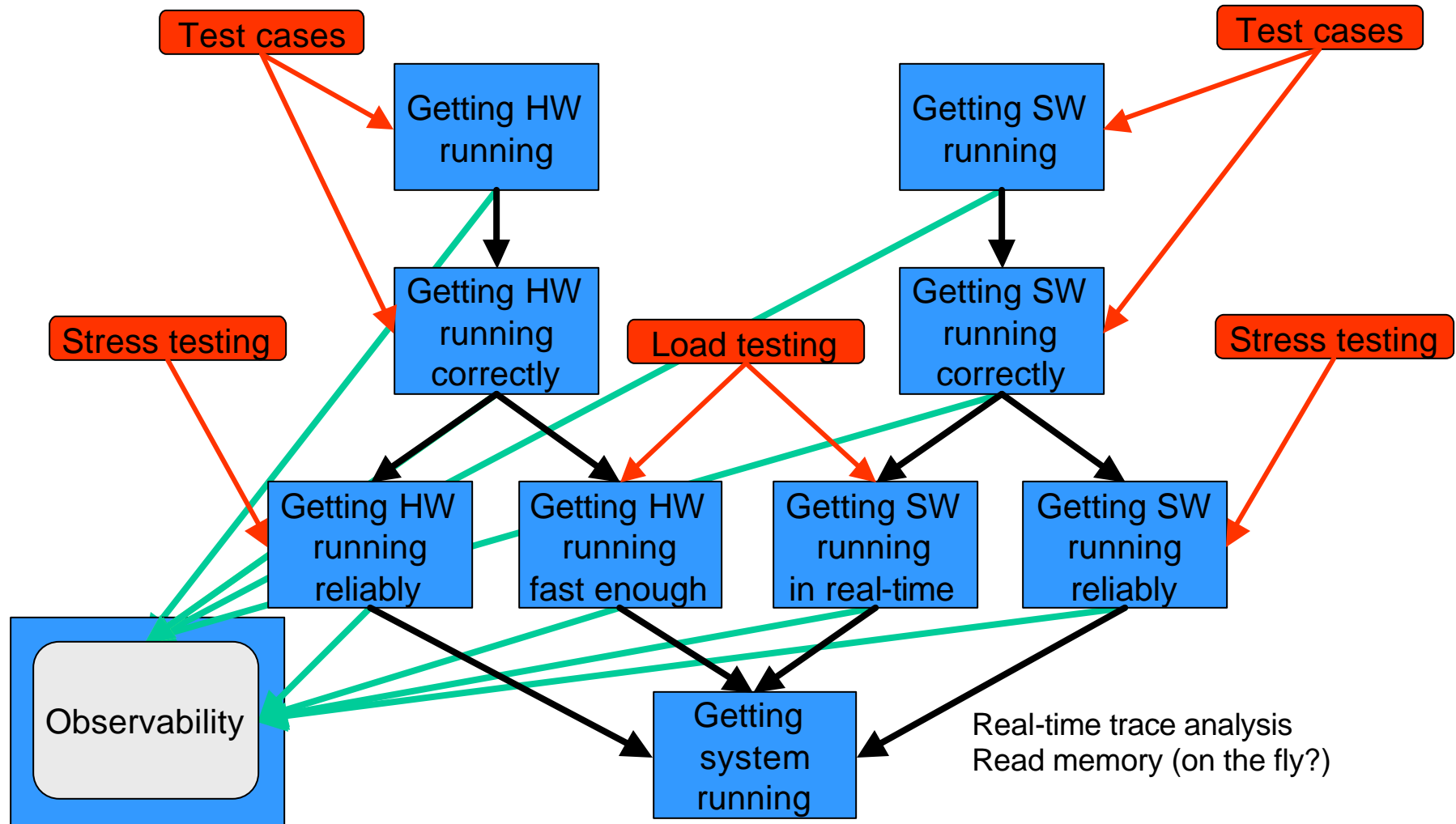


# Benchmarking performance

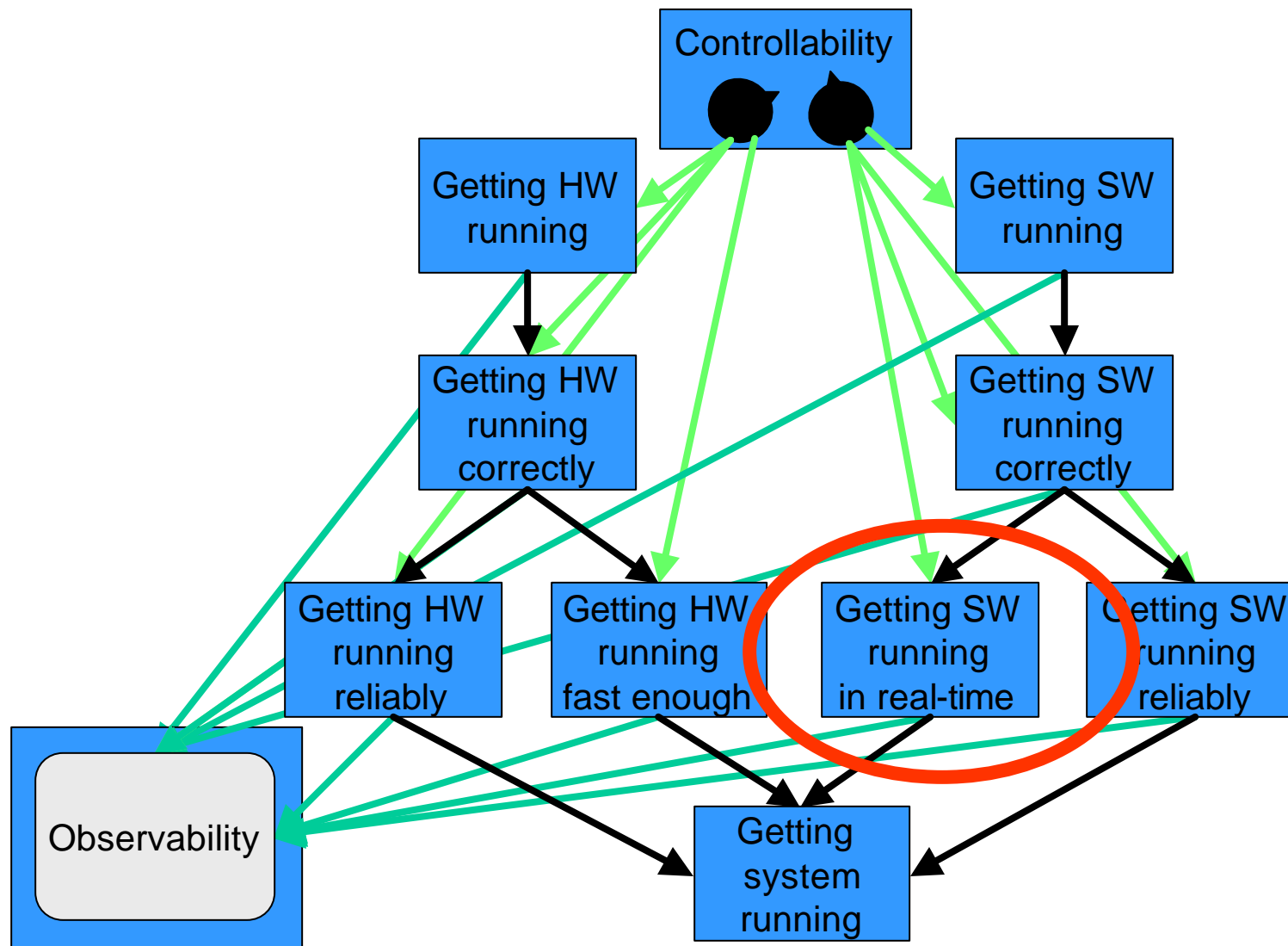
- Does everything get accessed?
- Is the benchmark representative?



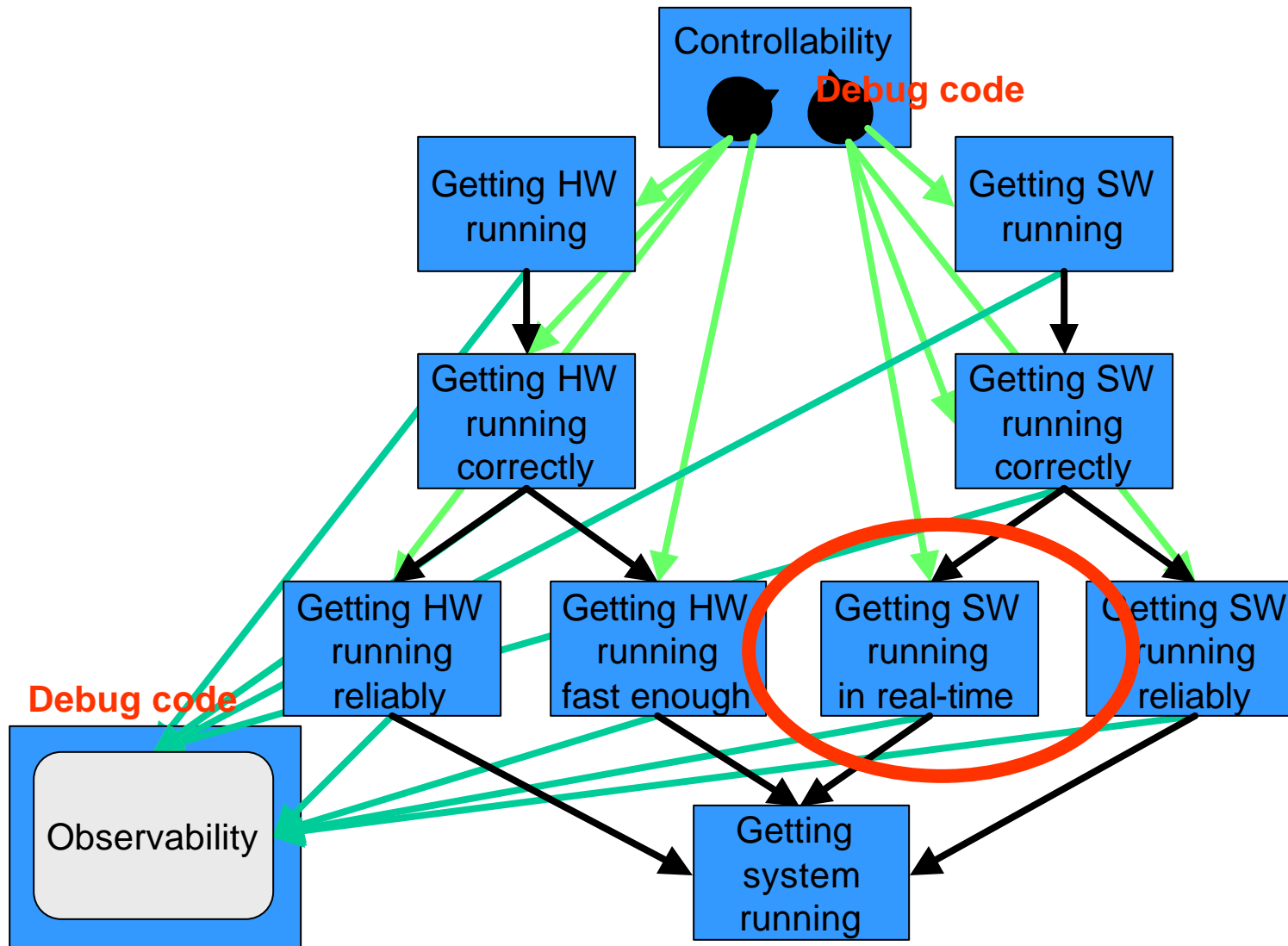
# Getting a Real-Time Embedded System Running



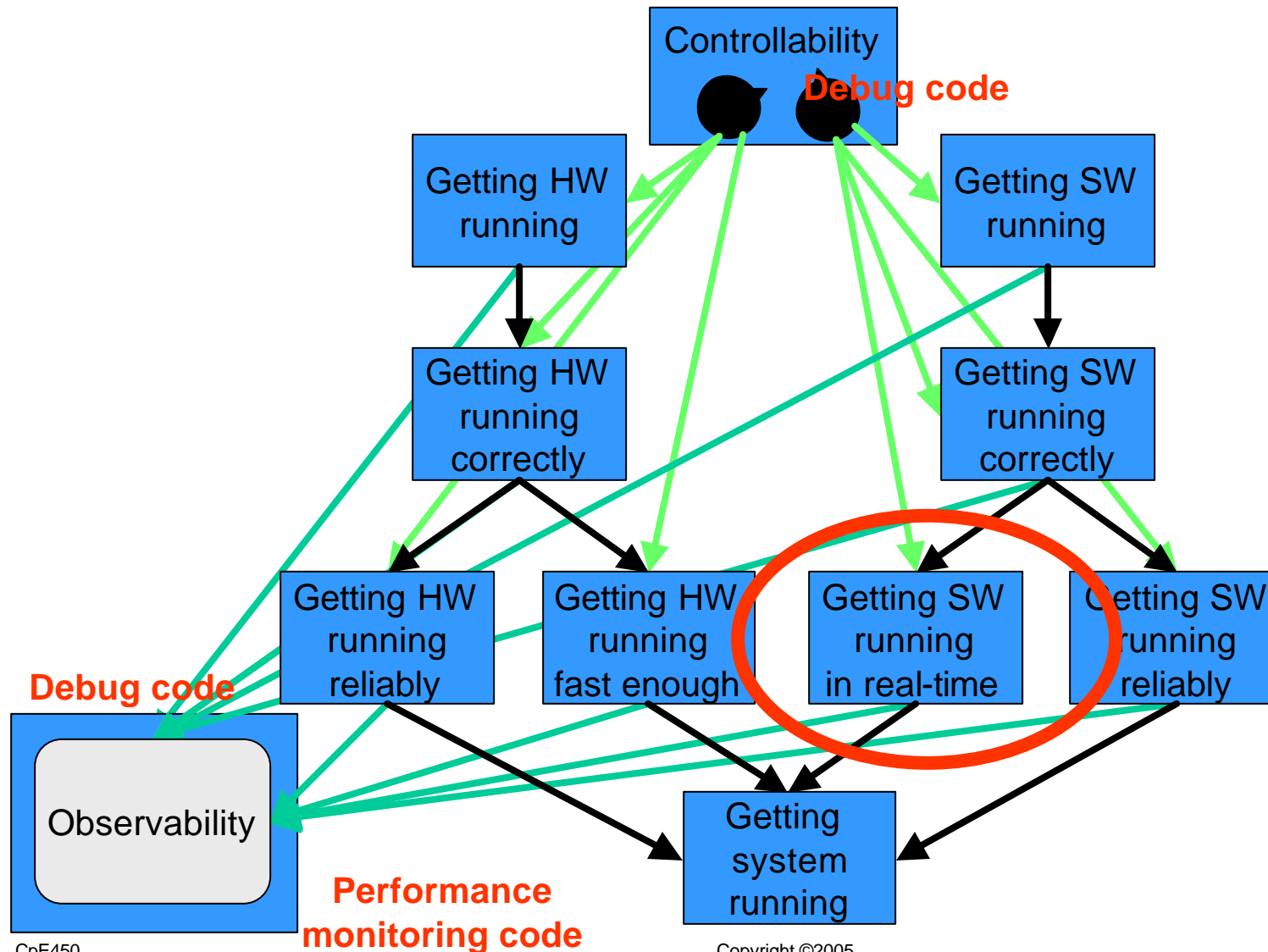
# Getting a Real-Time Embedded System Running



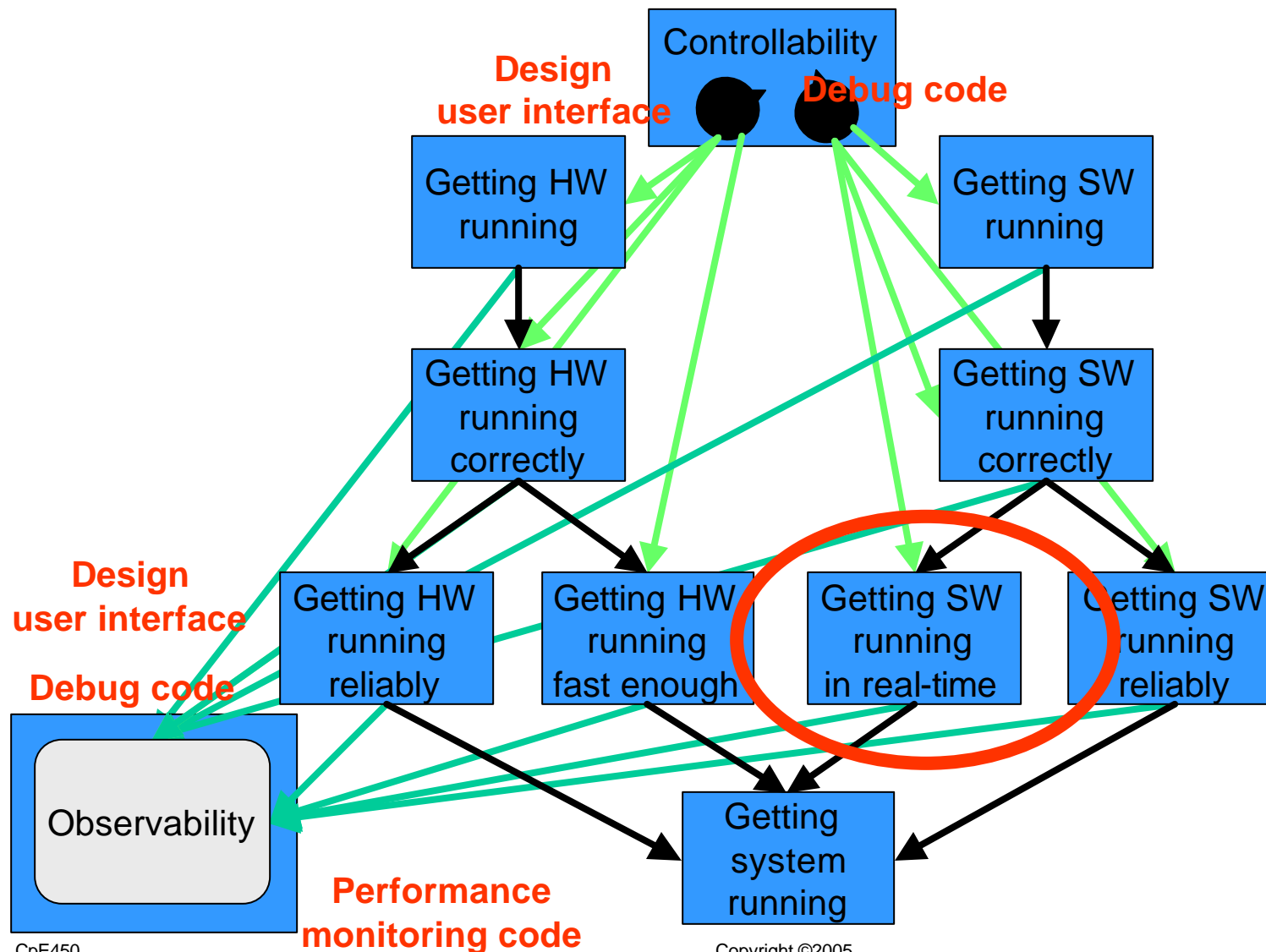
# Getting a Real-Time Embedded System Running



# Getting a Real-Time Embedded System Running



# Getting a Real-Time Embedded System Running



# Is Real-time *Monitoring* of Real-time *Performance* Necessary?

```
void main( )
{
  /* initialization code */
  while(1)
  {
    get_data(bits);
    generate_baseband(bits, sig);
    set_output_flag(1); ←
    modulate(sig, mod, BLOCK_LENGTH, Fc, delta_t);
    set_output_flag(0); ←
    output_mod(mod);
    sleep( );
  }
}
```

Real-time observation

# Is Real-time *Monitoring* of Real-time *Performance* Necessary?

```
void main( )
{
  /* initialization code */
  start_clock();
  while(1)
  {
    t_get -= clock();
    get_data(bits);
    t_get += clock();
    t_generate -= clock();
    generate_baseband(bits, sig);
    t_generate += clock();
    /* set_output_flag(1); */
    t_mod -= clock
    modulate(sig, mod, BLOCK_LENGTH, Fc, delta_t);
    t_mod += clock();
    /* set_output_flag(0); */
    t_out -= clock();
    output_mod(mod);
    t_out += clock();
    sleep( );
  }
}
```

# Is Real-time *Monitoring* of Real-time *Performance* Necessary?

```
void main( )
{
  /* initialization code */
  start_clock();
  while(1)
  {
    t_get -= clock();
    get_data(bits);
    t_get += clock();
    t_generate -= clock();
    generate_baseband(bits, sig);
    t_generate += clock();
    /* set_output_flag(1); */
    t_mod -= clock
    modulate(sig, mod, BLOCK_LENGTH, Fc, delta_t);
    t_mod += clock();
    /* set_output_flag(0); */
    t_out -= clock();
    output_mod(mod);
    t_out += clock();
    sleep( );
    /* periodically report the values of the t_ values */
  }
}
```

t\_get accumulates the total  
time spent in get\_data( )