Tickit: An Event-Based Multi-Device Simulation Framework

A. Emery, G. O'Donnell, C. Forrester, T. Cobb, Diamond Light Source, Harwell Science and Innovation Campus, Didcot, OX11 0DE

Tickit is an event-based multi-device simulation framework providing configuration and orchestration of complex simulations. It was developed at Diamond Light Source in order to overcome limitations presented to us by some of our existing hardware simulations. With the Tickit framework, simulations can be addressed using a compositional approach. It allows devices to be simulated individually while still maintaining the interconnected behaviour exhibited by their hardware counterparts by modelling the interactions between devices. Devices can be collated into larger simulated systems providing a layer of simulated hardware against which to test the full stack of Data Acquisition and Controls tools.

MOTIVATION

The Tickit framework was developed to facilitate the simulation of hardware triggered scans. Simulating scans of this nature requires multiple devices to communicate with one another and have linked behaviour triggered by events.

Our existing simulations consisted of devices created with the Lewis simulation framework and a number of stand alone device simulations. However, these grew increasingly complicated and verbose in order to provide the functionality required. This produced an increasingly fragile system which was unable to support further developments for the features we required. This drove the decision to develop a framework more appropriately suited to our needs. To do so a framework would need to be:

- Multi device - To simulate a full scan we need to have many linked devices.
- Event driven - Each device in the system needs to only update when relevant, either when it changes or when a device downstream changes an input to it.

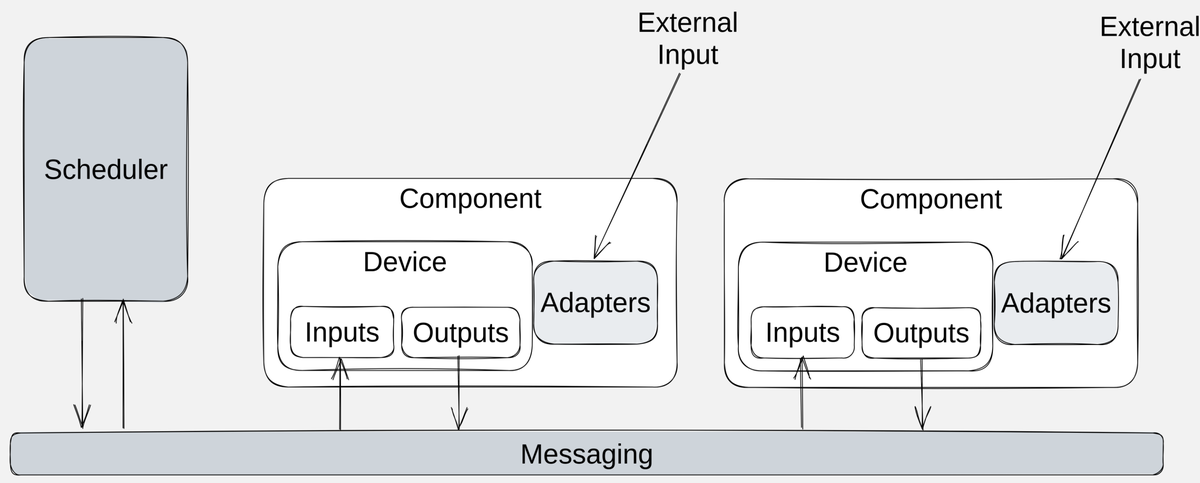
The scans we wish to simulate require use of a Zebra, an Eiger, and a PMAC. To support scans of this nature, we would need to simulate these devices as well as numerous motors and the communication between them. A significant limiting factor in our hardware triggered scan simulations was the high FPGA frequency. The Zebra FPGA operates at 50Mhz, a rate unachievable in a time driven system. Even with the ability to match this rate, using a time driven approach would drive the system intensely with the majority of these updates being redundant. As the simulation progressed, simulation time and real time would slowly diverge, the rate of the divergence increasing with added complexity. By using an event driven approach instead we only need to update each device when there is a relevant change. This enables the simulation to be run at a slower overall rate, lagging behind when updates are made, but then synchronising back to real time when there are periods of no change.

DESIGN

Scheduler

The scheduler orchestrates the running of the simulation. It contains references to all the components in the simulation as well as the wiring of all of their inputs and outputs. It is responsible for routing all the changes through the system and ensuring time is maintained.

The Tickit framework consists predominantly of two parts: a scheduler, and components.



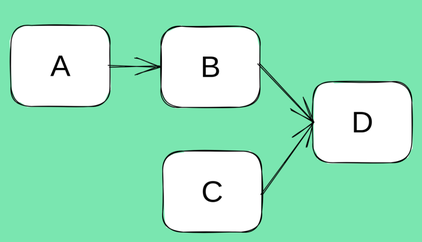
Components

Components encapsulate the simulated devices. They provide the interface for the running, updating of and interactions between devices. Components act as the interface between devices and the scheduler.

There are two types of components in Tickit; device components and system components.

Multi-device

All devices possess optional inputs and outputs, which can be wired together to produce a directed acyclic graph of dependent devices. This device graphing is depicted using a configuration yaml file which is used to build and run the simulation.

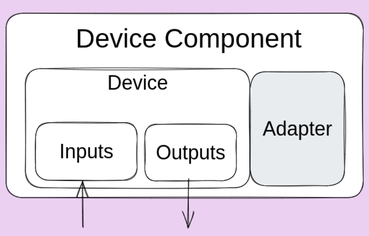


The scheduler ensures devices are updated in order. In the above case it ensures B is only updated when A has finished, and D is only updated once both B and C have finished.

Adapters

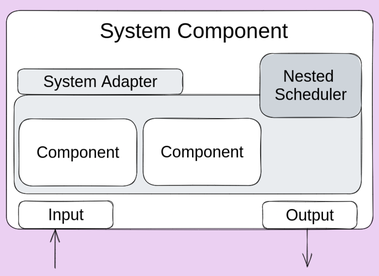
Adapters control and read data from devices using a network interface. They were designed to provide a simple way for users to write device specific interfaces while avoiding re-implementing standard I/O logic. Each device needs a specific implementation of a given adapter which will be paired with the required I/O to allow the adapter to be run within the component.

Device Components



Device components are the predominant component type and encapsulate the simulated device and any adapters for that device. The devices can be considered directly linked with regards to inputs and outputs.

System Components



System components are a component type that nest a Tickit simulation within them. They contain a set of components, a nested scheduler for routing the updates through the components, and an optional adapter for maintaining an overview of the nested system.

TICKIT DEVICES

Devices are the user implemented code for the simulation. There is a minor amount of required boilerplate for these devices. They must contain input and output nested classes and an update method which produces a device update. The inputs and outputs are required for device graphing, and the update method contains the user implemented code for how the device should behave when one of its inputs is changed.

We have started development on two of the main devices needed; a Zebra and an Eiger.

DEVELOPMENTS

Going forward, our efforts will be focused on developing the Zebra device and providing any framework changes required to facilitate it. The most significant of these will be the introduction of re-wirable graphs within system components needed for live block manipulation within the Zebra.

Alongside this there will be efforts extend the functionality of the Eiger and to produce an initial simulated PMAC device. Preliminary work has been done to explore the requirements of a PMAC simulation however further work needs to be done to produce a working device.

There are also goals to enable the distributed use of Tickit simulations. Tickit has been designed to allow the scheduler and components, and any subset thereof, to run in different processes. This allows us to make increasingly complex simulations by distributing the operational load. It also would allow us to create containerised deployments of sets of devices, configured for specific beamlines.

Achieving these goals will provide us with the devices and features required to further improve and expand our beamline simulations. This in turn will provide us with greater testing capabilities and be a valuable asset in the development, maintenance and debugging of our acquisition and controls tools.

For more information please visit www.diamond.ac.uk or contact Abigail Emery at abigail.emery@diamond.ac.uk