Software Development with Real-Time Workshop Embedded Coder

Nigel Holliday
Thales Missile Electronics

Contents

- Who are we, where are we, what do we do
- Why do we want to use Model-Based Design
- Our Approach to Model-Based Design
- Where did we use Model-Based Design
- What benefits were seen
- What difficulties did/do we experience
- Where do we want to go now
- Conclusions so far

Who are we - Thales UK Operations in Defence

- 2nd largest defence systems contractor in the UK
- Operates at 3 levels in the UK market
  - Prime contractor
  - Sub-system integrator – where we take responsibility for integrating complete sub-systems for a platform
  - Sub-system supplier – where we will offer in competition world class technology and / or products
- Building on our core systems integration capability
- Growing CLS (Customer Logistic Support) business

THALES

- TME: Basingstoke
  - Single Integrated Site
    - On-site manufacturing
    - Laboratories
    - Environmental test facilities
    - 240 staff

What do we do


Weapon Systems
Safety & Arming Units; Hard Target Fuzes
Proximity Fuzes / Target Detection Devices
Multimode Seekers & Sensors
Systems, Technologies & Products
Battlefield Systems
Battlefield Target Identification, UAS Sensors & Payloads
Consultancy
ILS, CLS, Rad Haz Consultancy, Telemetry & Weapon Range Support, Research, Security Scanner

Why do we want to use Model-Based Design

- Save money!
  - Reduce coding effort and timescales
  - Reduce introduction of errors – reduced risk
- Reduce the need for documentation
  - Requirements - DOORS
  - Design specifications – lost in translation!!
    - The model is the design – graphical solution but well documented

Why do we want to use Model-Based Design

- Rapid prototyping
  - Early checking of software on target - timing/resources
    - Functional correctness of algorithms
    - Determine run-time and memory requirements
  - Design decisions on target hardware
    - Put on eval boards quickly to confirm following
      - 16-bit or 32-bit
      - Floating or fixed point?
      - Memory – internal/external?
      - FPGA required?

Why do we want to use Model-Based Design

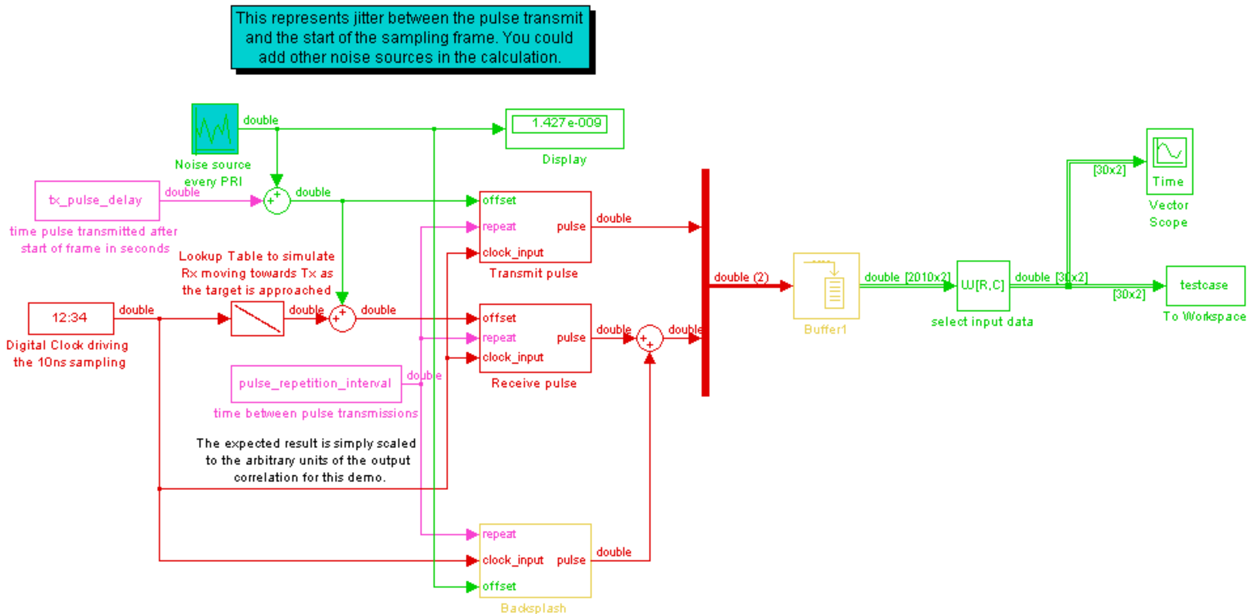
- More efficient use of resources
  - Modelling engineers concentrate on creating the model and supporting real-world environments
  - Embedded engineers concentrate on processor scheduling and I/O to the rest of the physical system
    - The model plugs into the embedded software harness
  - Uptake of Model-Based Design could lead to less distinction between the two disciplines
    - Increased labour flexibility – common toolsets
    - Hybrid engineers!!
    - Broader understanding of design and implementation

- TME approach to Model-Based Design was not to use it in the harness
  - Decision at the start of the pilot project was the model was to plug into a hand-coded scheduler/harness
  - C coding was used for all software programming of the target resources
  - Model could be taken from the Simulink “real-world” environment and C code generated
    - Some processor I/O simulation in real-world environment where required

- Two projects used MBD
  - P1: Data processing for a single channel pulsed proximity sensor + timing algorithm
    - TME designed custom hardware for TDP
    - Software developed for 2 x dual-core 16-bit fixed-point DSPs
      - Serial and parallel I/O required with DMA
      - FPGA + analogue front-end
  - P2: Control algorithms for a gimbal assembly with mounted pulsed laser and PIR dual mode sensing
    - COTS hardware with 4 x floating-point DSPs
    - Single DSP used to run model
      - Parallel I/O
      - FPGA – gateway to rest of the system
    - Vendor board support library

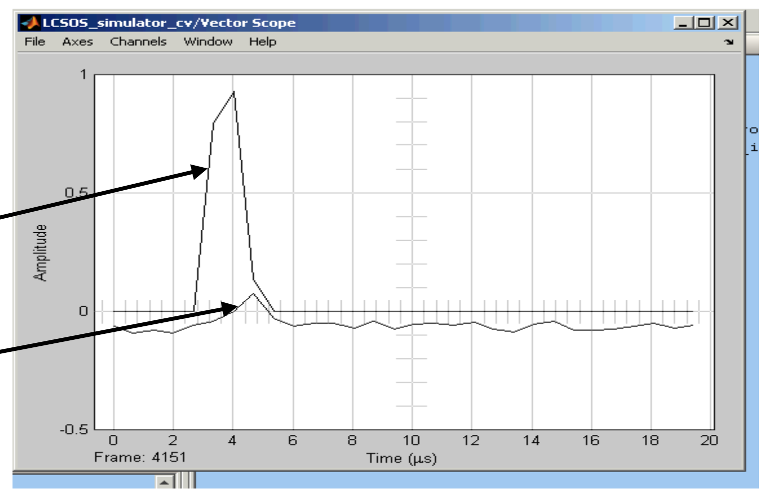
Where did we use Model-Based Design: P1

Simulator



- Create representative simulator
- Historic Information
- Use measured results

Data saved as .mat file

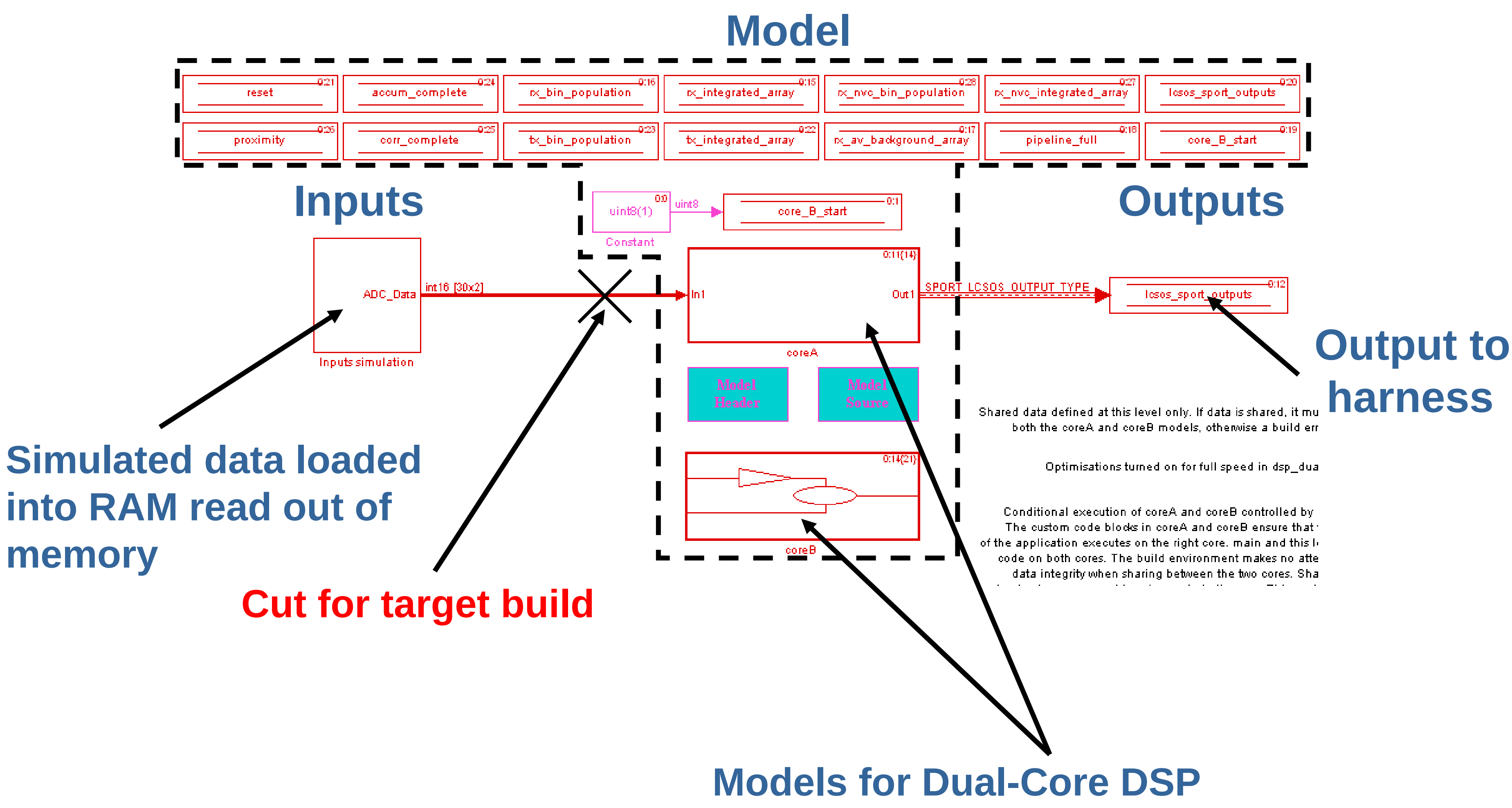


Tx pulse with noise

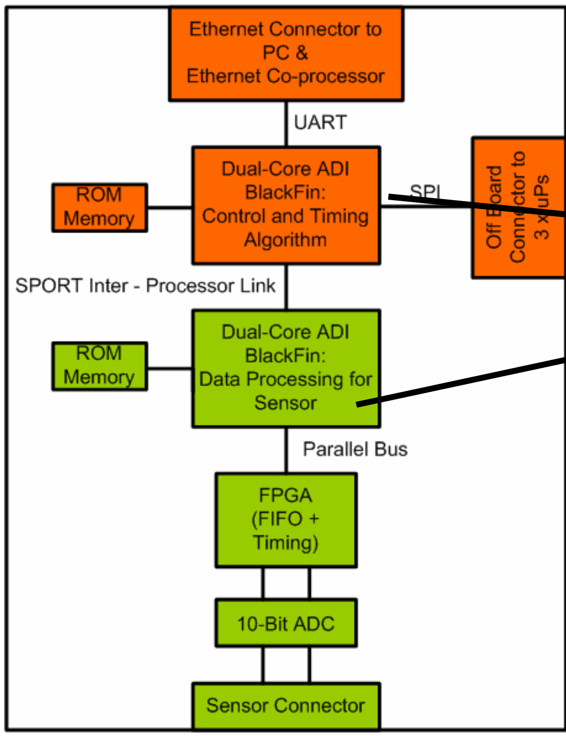
Rx pulse with noise & range law

THALES

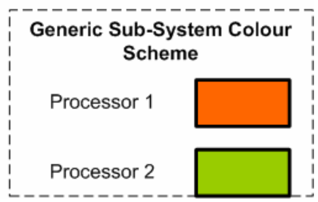
Where did we use Model-Based Design: P1

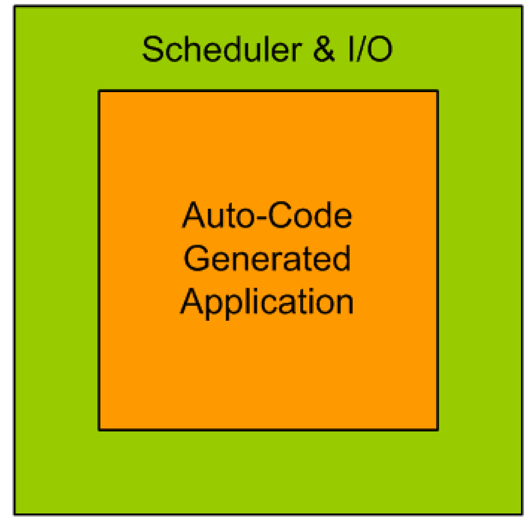


Where did we use Model-Based Design: P1



Simple Ideology



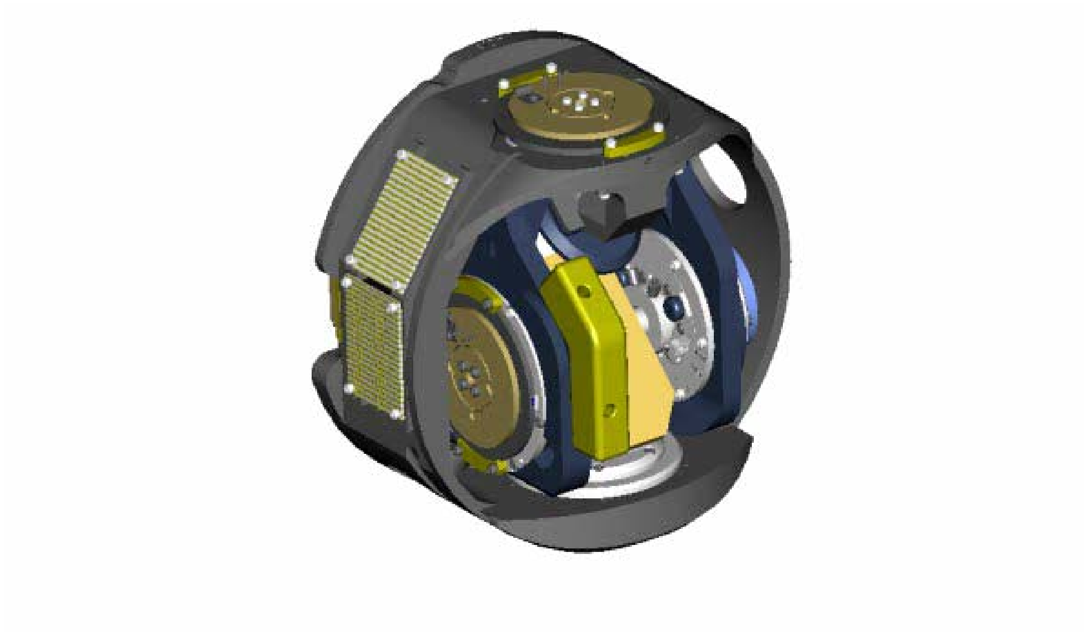


Generic scheduler with I/O for processor family

TME Custom Hardware

Where did we use Model-Based Design: P2

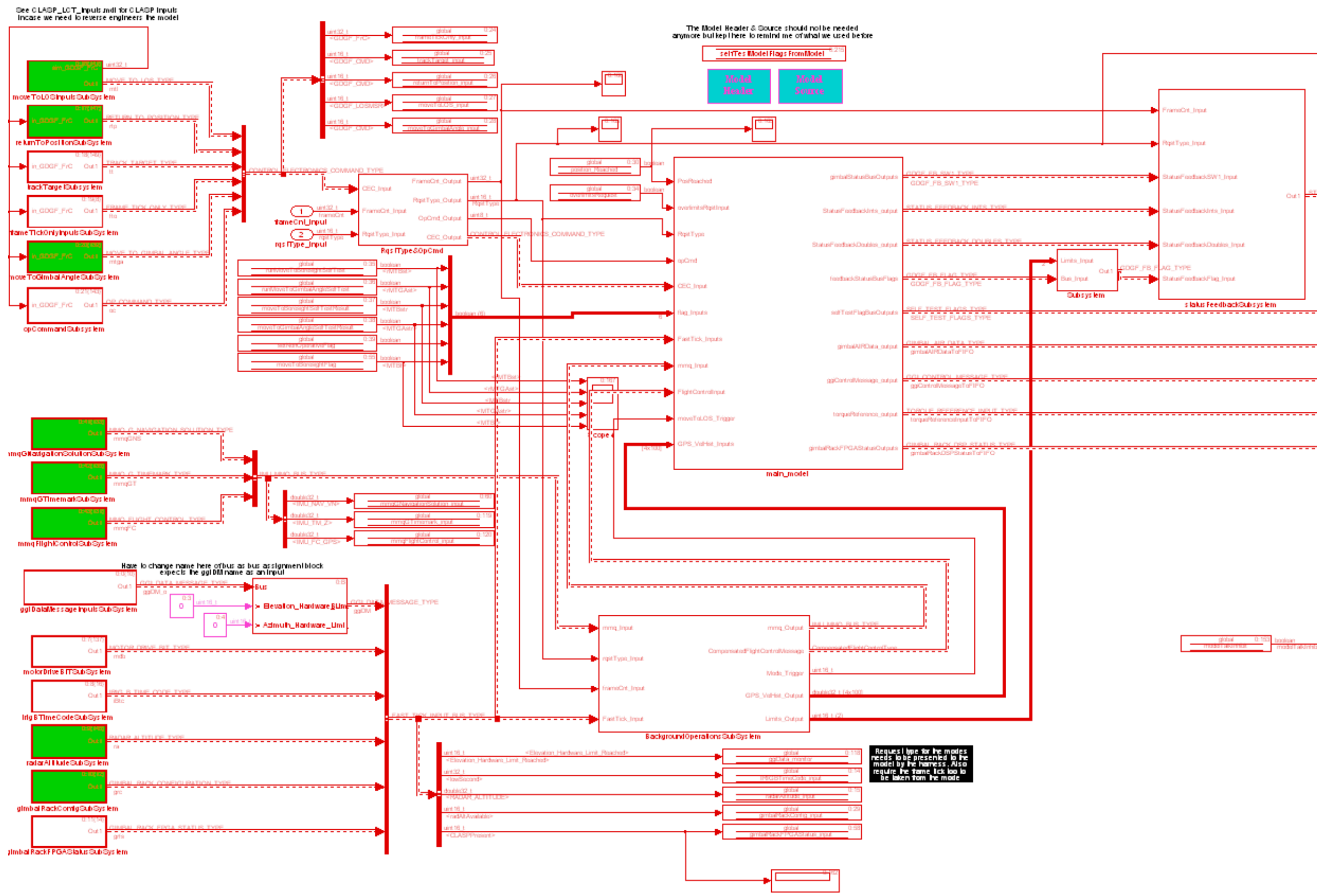
- Simulation
  - Real-world model in Simulink
    - Several modes required
    - Single mode simulation model optimal– time/cost v payback
  - Gimbal model developed in ProE



Where did we use Model-Based Design: P2

- Inputs derived from real-world model
- Model evaluated on hardware and compared against simulation for timing & correctness – it does what it says on the can

Inputs | Model | Outputs

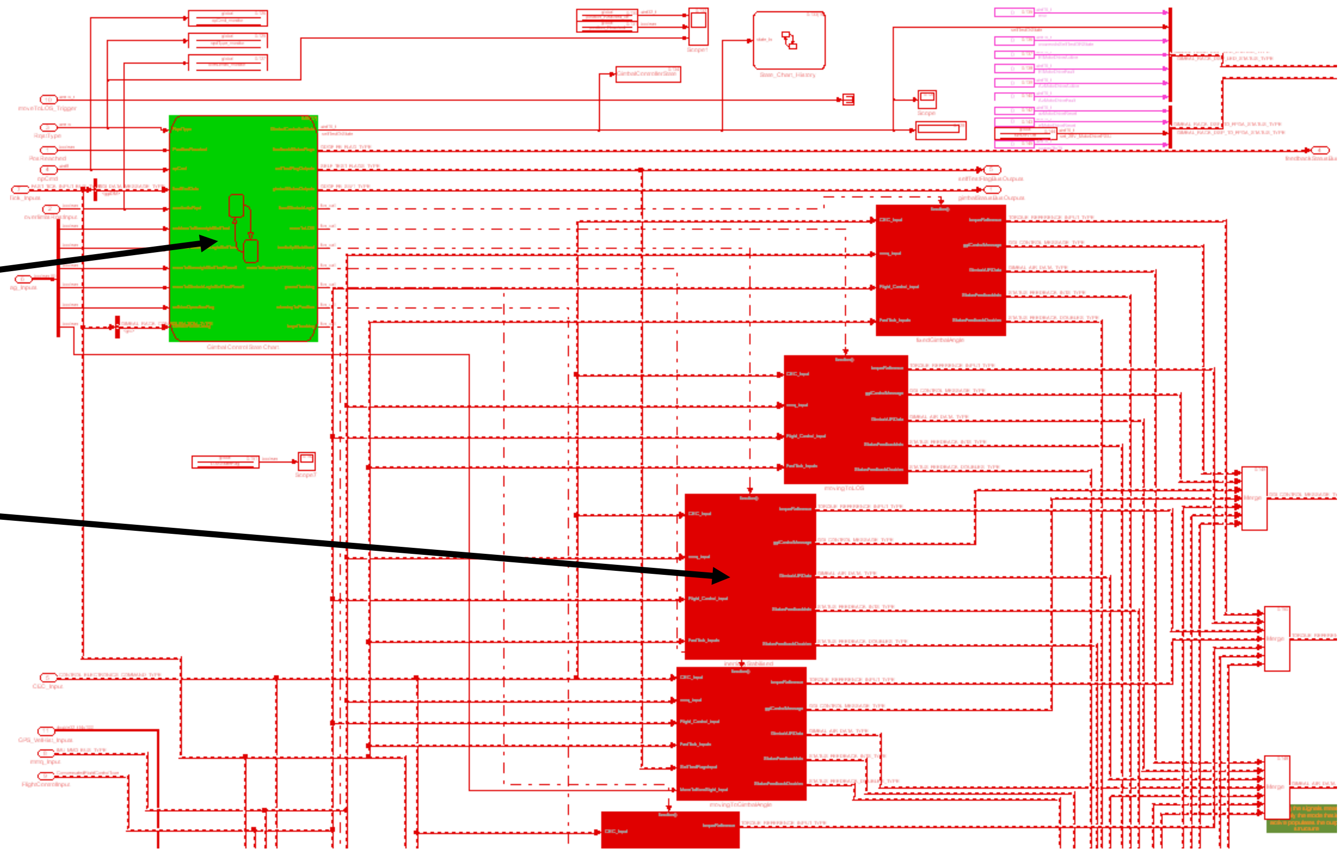


Top Level Model

Where did we use Model-Based Design: P2

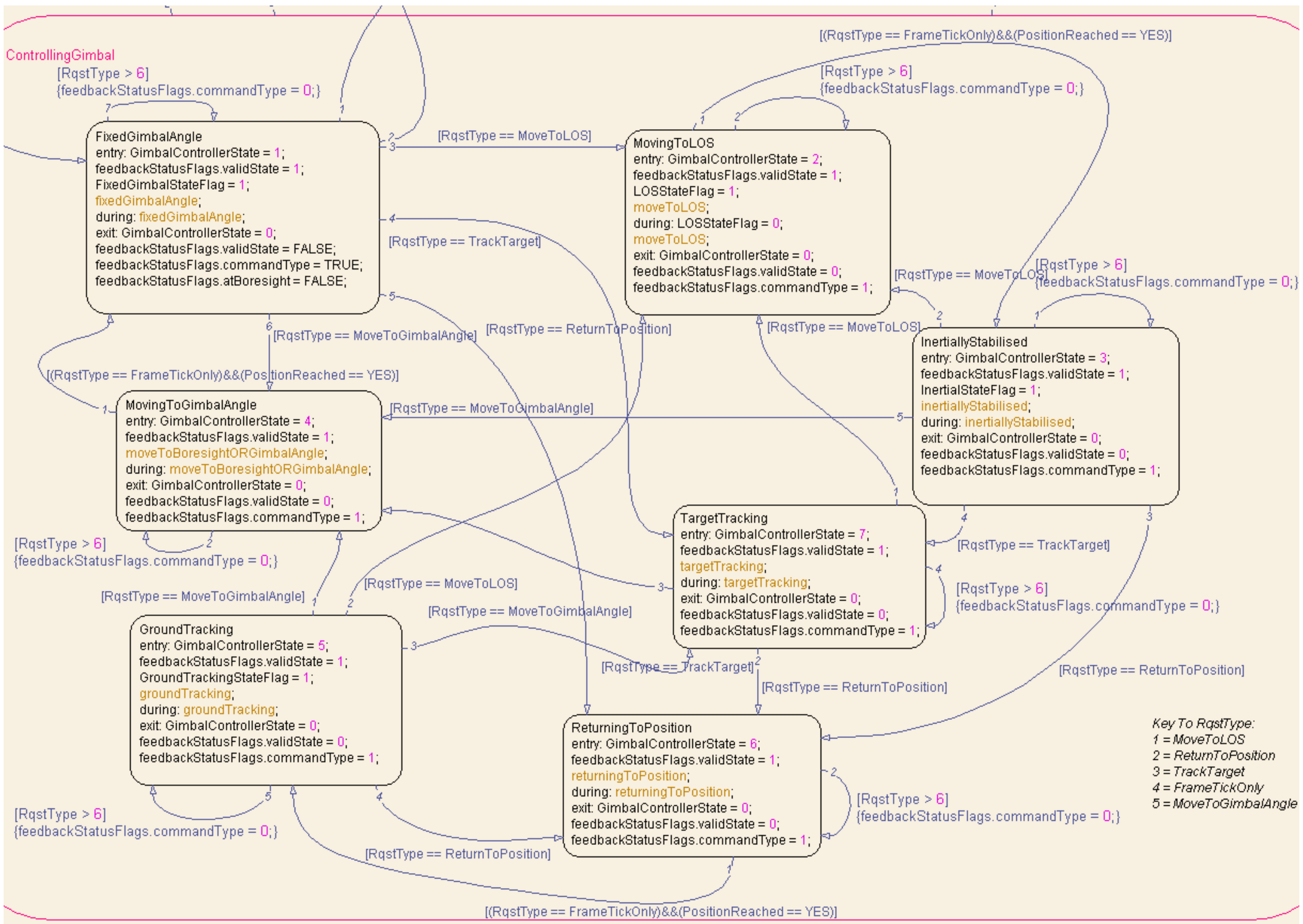
Second Level Model

- State-machine implemented in state-flow
- Modes/States picked from original simulink model



Where did we use Model-Based Design: P2

Gimbal State Controller



- Re-use of simulation data
  - Same stimuli used for model verification on hardware
    - Easy/fast capture of test stimuli for model from real-world model
    - Cross referencing simulation and hardware model versions
- Rapid prototyping possible
  - Extensive use of low cost microprocessor evaluation boards prior to making hardware decisions
  - Evaluate model and hardware it is to run on
  - Timing analysis/profiling – can the model run fast enough on hardware
  - Optimise parts of model if necessary

- Reduced specification writing
  - No need for lengthy detailed design specs
    - Well documented model with graphical flow can yield almost as much detail as a written specification – can do this in the model
    - Well organised model with several tiers can clearly show model hierarchy (with adequate labelling)
    - Software interface documentation still required
- Rapid response to change/additions to requirements
  - New model sections rapidly integrated and tested on hardware
    - Maximise use of existing architecture – greater visibility with graphical model

- Powerful linkage between model and software run on the hardware established
  - During integration can return easily to model for debug
    - Simulink display facilities allow easy visibility for rapid debug
    - Still use microprocessor development environment
      - Breakpoints
      - Memory/register contents
      - Execution time
    - Can aid debug of third party sub-systems
- No perceivable increase in development time during the learning curve period
  - Scheduler required significant development time
  - This needs to be done anyway

- Ability to review model with third party
  - TME program management team
  - Customer
  - Other team project members
  - Internal review processes

- Where to start!!!
  - No prior experience of Simulink or Stateflow
    - Mathworks training courses only in 2005
  - How to architecture the model for simulation
  - Limited experience of house keeping activities for code generation from a Simulink model
    - Template Make Files
    - Low level understanding of compiler options
    - Code and data placement in memory

- Pressures to deliver on a live project
  - Learning curve to go up
- Debugging the model
  - Setting breakpoints in the model
  - Is it Simulink or the target environment
  - Program flow through the model
    - Graphical interpretation of execution order
    - Program control sometime difficult to understand
    - In-built debugger hard to drive – lack of training/experience?

- How to configure a model for multiple developers
  - TME uses Sourcesafe for software
  - How do we handle multiple developers on a single model for configuration and integration – even for desktop development
    - More acute for embedded applications

- Demonstrate significant reductions in timescales for model based development
  - Acceptance by program managers and company hierarchy only if visible savings
- Define a company process for model based design involving code generation
  - Record current knowledge so not lost!
  - Iterative/learning process
- Use on more projects
  - Increase expertise in model based design across the company product range and staff – where applicable

- MISRA compliant hand/model generated code
  - Future products expected to require safety related software
  - Increase documentation within the models
- Make use of linkage with DOORS
  - For bigger programs
  - Simplify requirements and compliance management
- Make more use of in-built Simulink reporting tools to better describe model – the model is the specification

- No perceived increase in development time/cost in early programs
  - Savings masked by other activities that are also on the learning curve – e.g. new processor
- If it happens in the model it will happen on the target
- Re-use of simulation data allows early evaluation of algorithms/models on target resources
- Model-Based Design very flexible and responsive to change (for example dual vs. single core)

- Still work to do to define a process
  - Iterative activity to get to a process that works
  - Flexible process to cater for desktop and embedded applications
- MathWorks pilot support throughout - Excellent!

- Similar pilot study evaluating Model-Based Design carried out at a Thales sister company in Belfast
  - Automatically generated fixed point code ran 30% faster than the hand written fixed point code