EE4212 Brick-breaker project

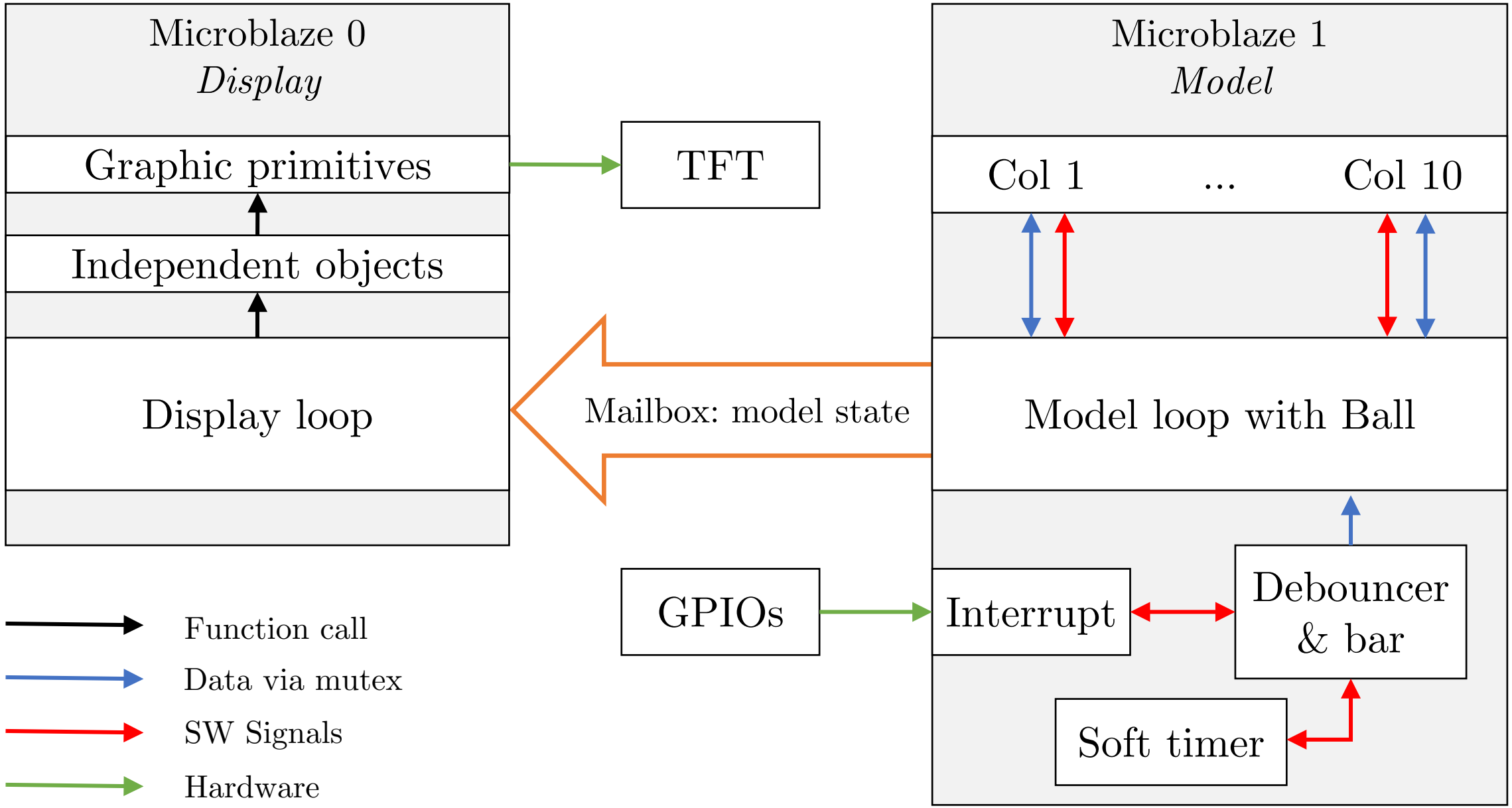
Real-Time Embedded Systems

Academic Year 2016-2017

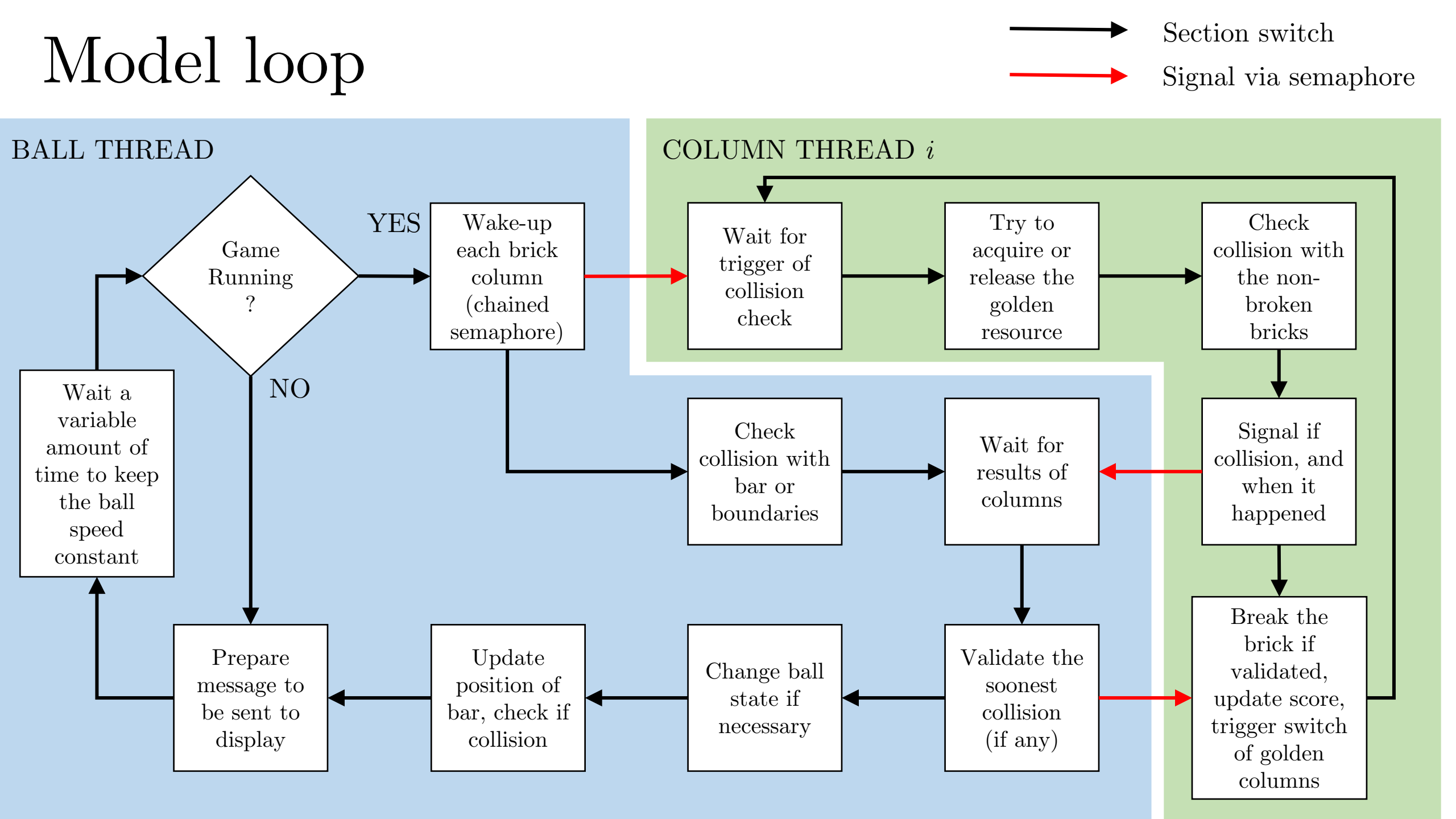
Paul-Edouard Sarlin

A0153124U

Overview


Microblaze 0
Display
Graphic primitives
Independent objects
Display loop
TFT
Microblaze 1
Model
Col 1
...
Col 10
Mailbox: model state
Model loop with Ball
GPIOs
Interrupt
Debouncer & bar
Soft timer
Function call
Data via mutex
SW Signals
Hardware

Model loop


Section switch
Signal via semaphore
BALL THREAD
COLUMN THREAD i
Game Running ?
YES
NO
Wake-up each brick column (chained semaphore)
Wait for trigger of collision check
Try to acquire or release the golden resource
Check collision with the non-broken bricks
Signal if collision, and when it happened
Break the brick if validated, update score, trigger switch of golden columns
Wait for results of columns
Check collision with bar or boundaries
Validate the soonest collision (if any)
Change ball state if necessary
Update position of bar, check if collision
Prepare message to be sent to display
Wait a variable amount of time to keep the ball speed constant

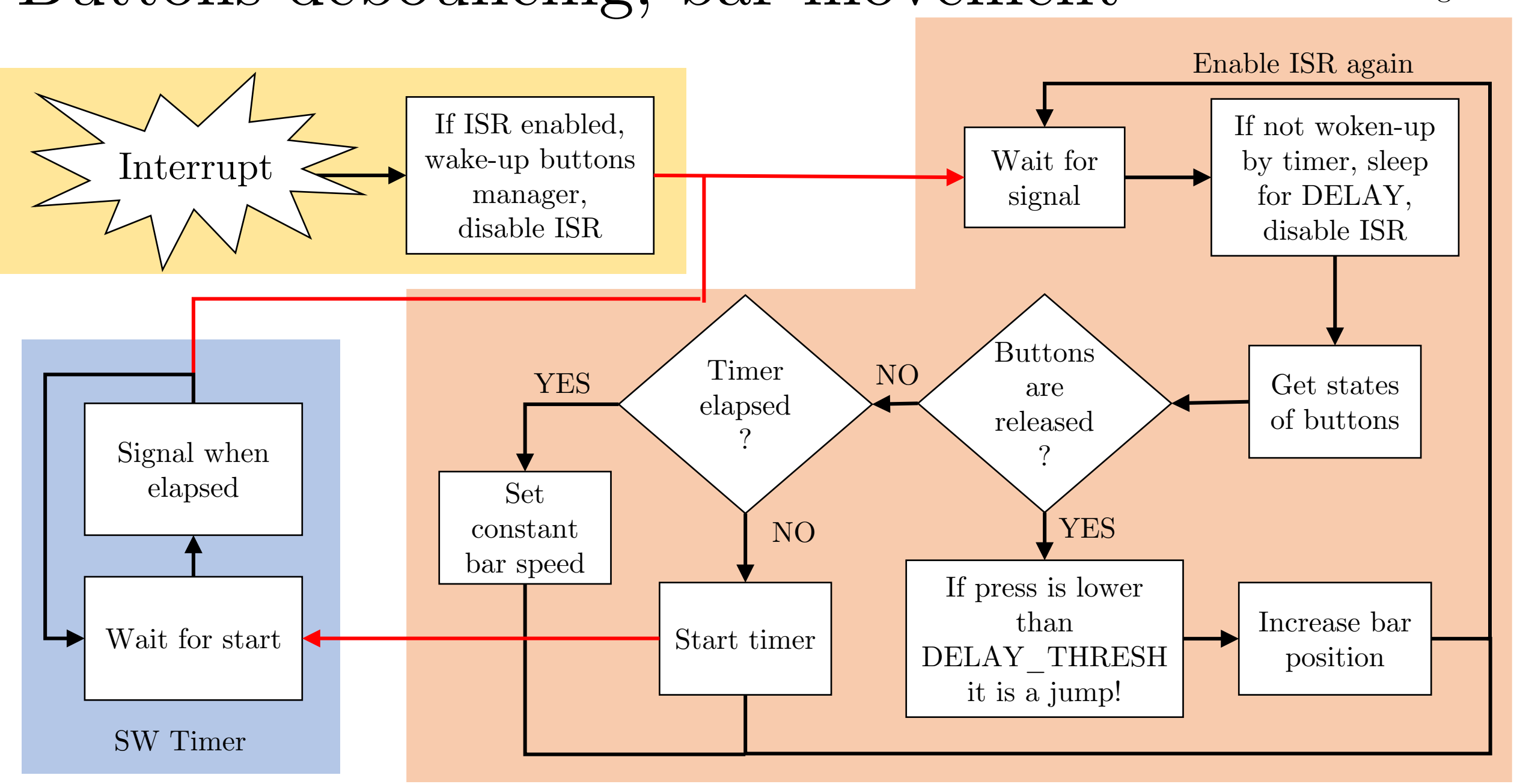
Collision checking for each column

```
If ball is not very far
   For each position between current and next frame:
      If ball is outside the zone of the column:
                  Move to next position
      For each non-broken brick of the column:
                  If ball is touching any edge or on any corner:
                        Compute normal of the corresponding surface
                        Send normal and corresponding iteration to ball
Signal to the ball that there is no collision
```

- In ball_thread:
  - Pick the lowest iteration = soonest collision
  - Send back the arbitration to the columns
  - Update ball state

Similar procedure for collision with bar/boundaries

Buttons debouncing, bar movement



Salient features

- High FPS: up to 33
- Coloured bar regions
- Pseudo-random golden pattern
- Random initial ball angle
- Random initial golden columns
- Ability to pause the game anytime
- Double frame-buffer -> no flicker
- Minimum coupling between uB0 and uB1
  -> Can reset any of them at anytime

Possible improvements

- Draw all of the objects concurrently (threads) -> can increase FPS
- Reset the game with a button (without resetting uB1)