L11: Major/Minor FSMs

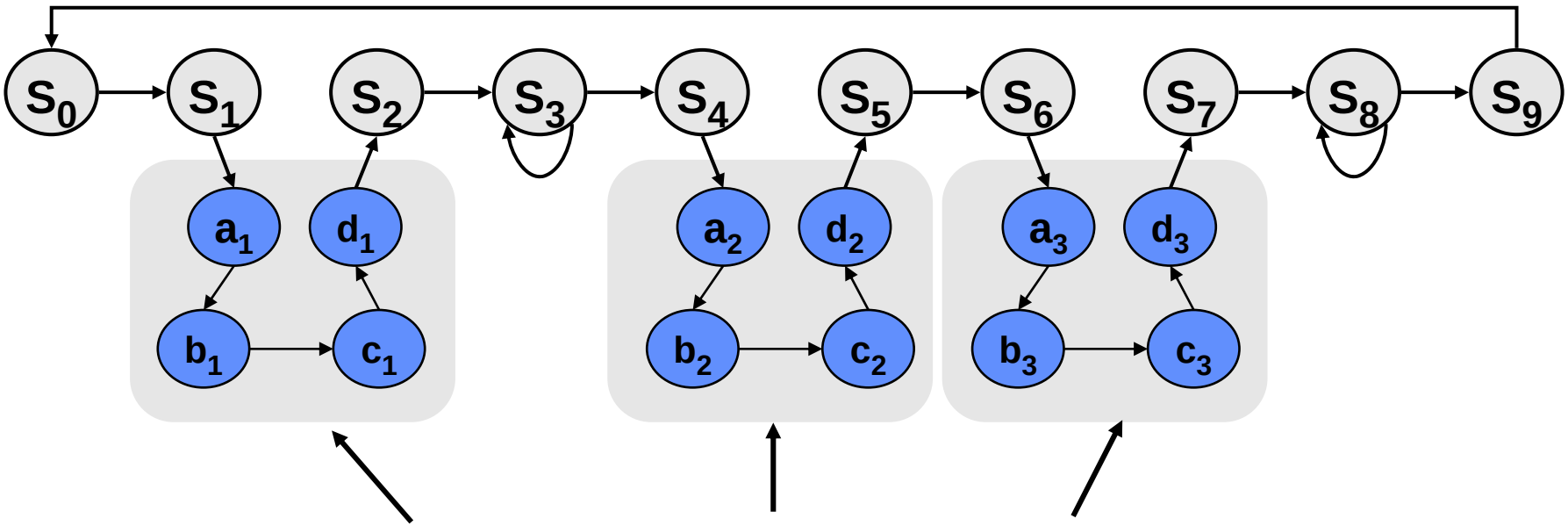
Acknowledgements:

Materials in this lecture are courtesy of the following sources and are used with permission.

Rex Min

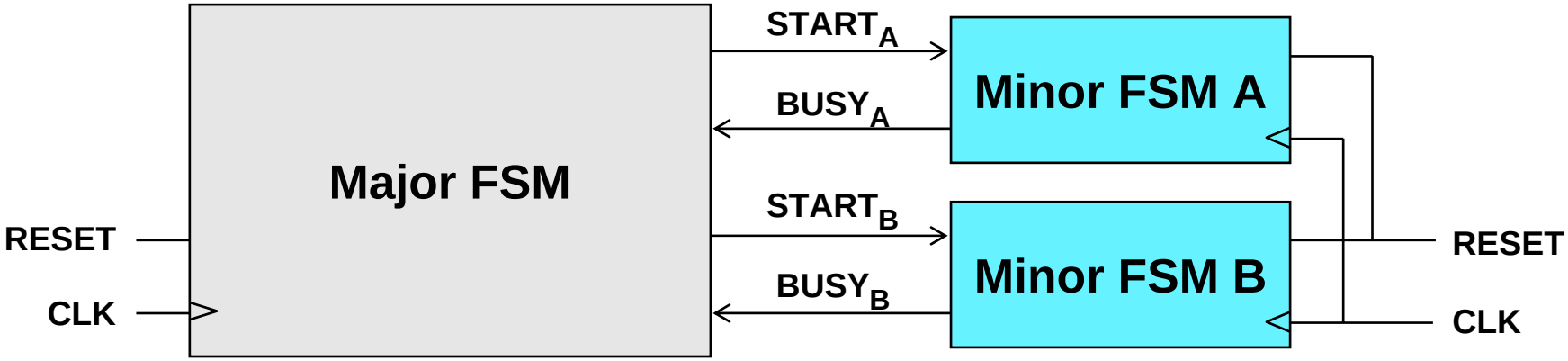
Toward FSM Modularity

- Consider the following abstract FSM:



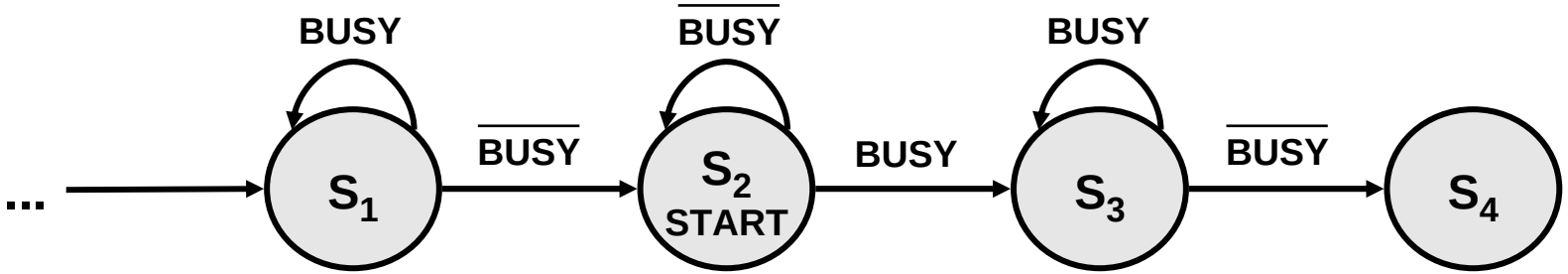
- Suppose that each set of states a_x...d_x is a "sub-FSM" that produces exactly the same outputs.
- Can we simplify the FSM by removing equivalent states?
  No! The outputs may be the same, but the next-state transitions are not.
- This situation closely resembles a procedure call or function call in software...how can we apply this concept to FSMs?

The Major/Minor FSM Abstraction



- Subtasks are encapsulated in minor FSMs with common reset and clock
- Simple communication abstraction:
  - START: tells the minor FSM to begin operation (the call)
  - BUSY: tells the major FSM whether the minor is done (the return)
- The major/minor abstraction is great for...
  - Modular designs (*always* a good thing)
  - Tasks that occur often but in different contexts
  - Tasks that require a variable/unknown period of time
  - Event-driven systems

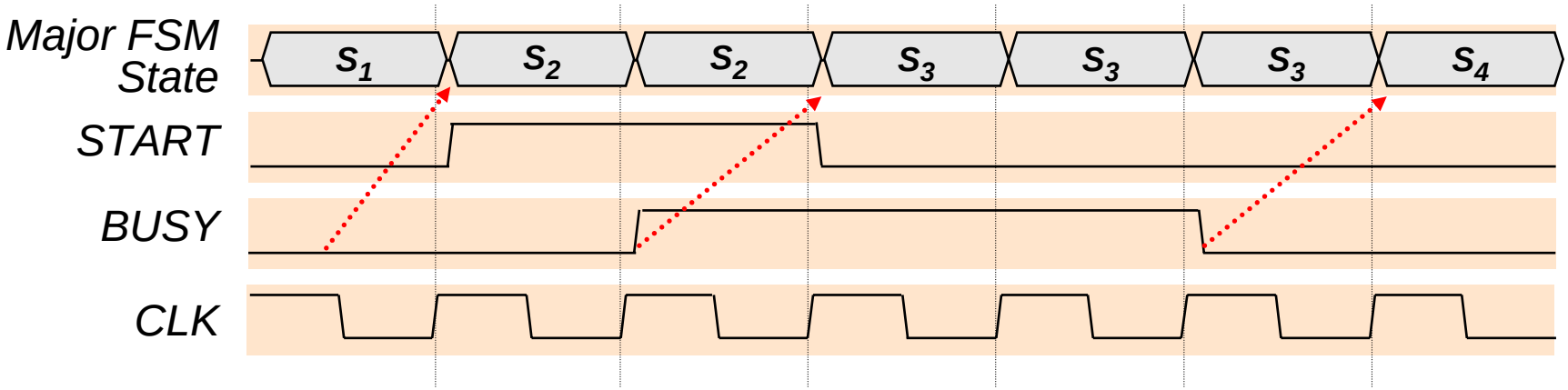
Inside the Major FSM



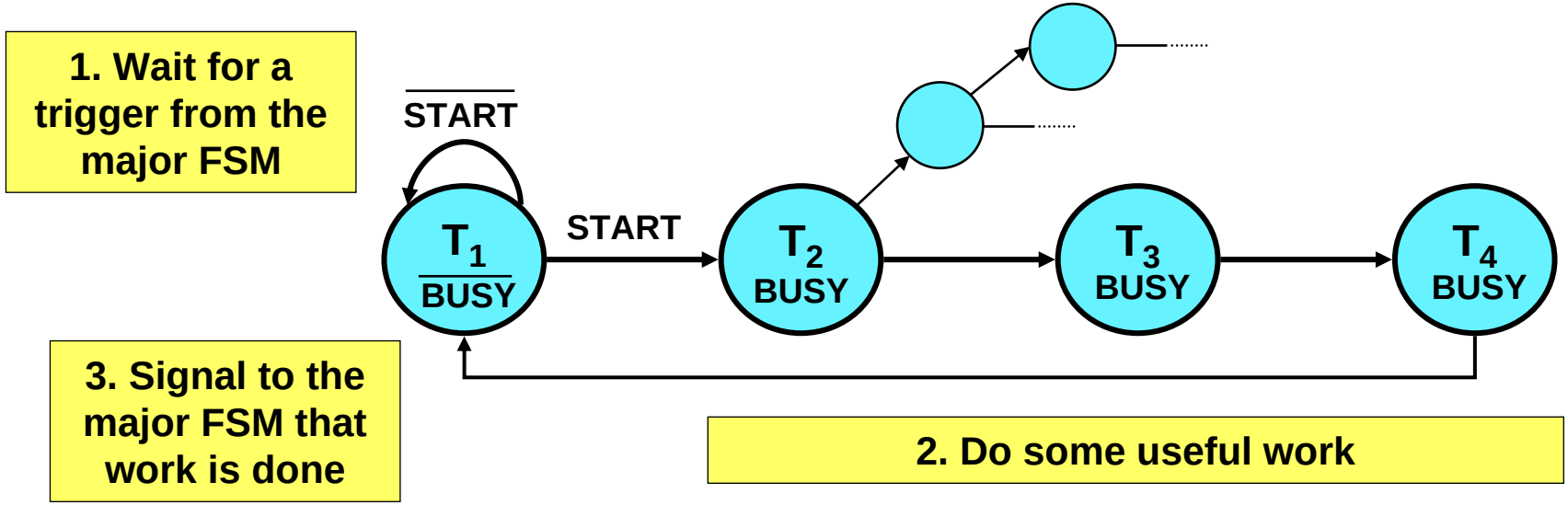
1. Wait until the minor FSM is ready

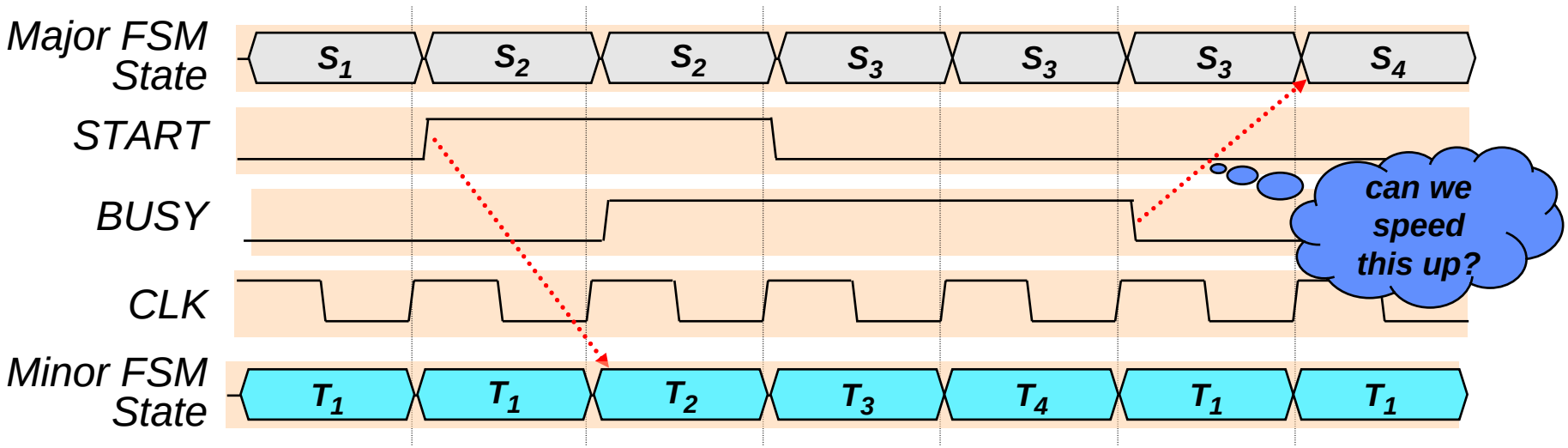
2. Trigger the minor FSM (and make sure it's started)

3. Wait until the minor FSM is done



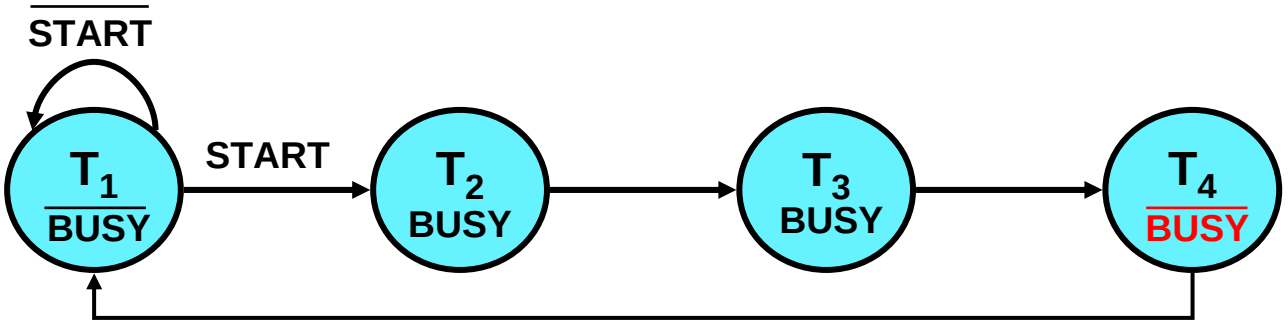
Inside the Minor FSM





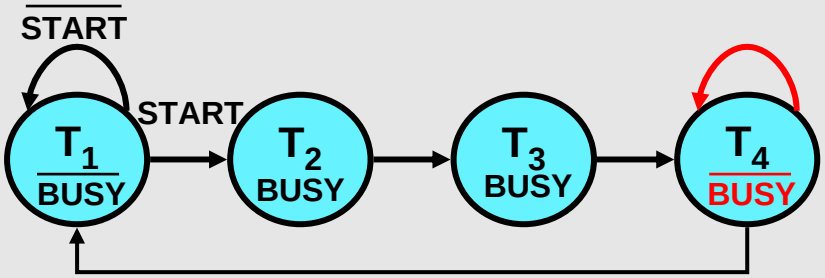
Optimizing the Minor FSM

Good idea: de-assert BUSY one cycle early



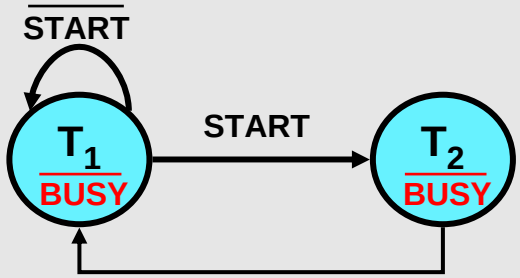
Bad idea #1:

T_4 may not immediately return to T_1

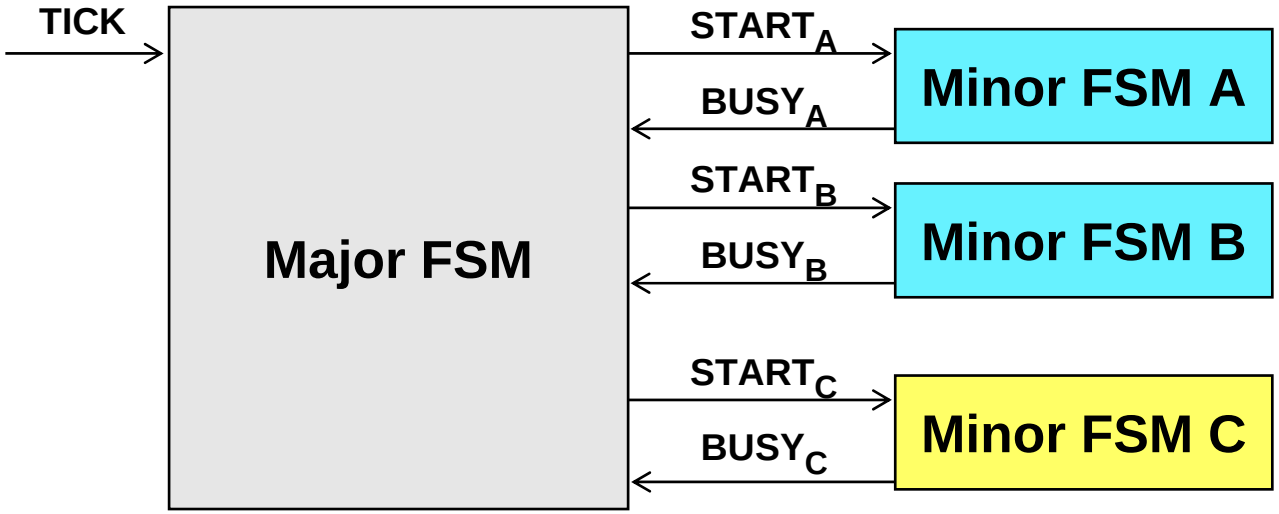


Bad idea #2:

BUSY never asserts!

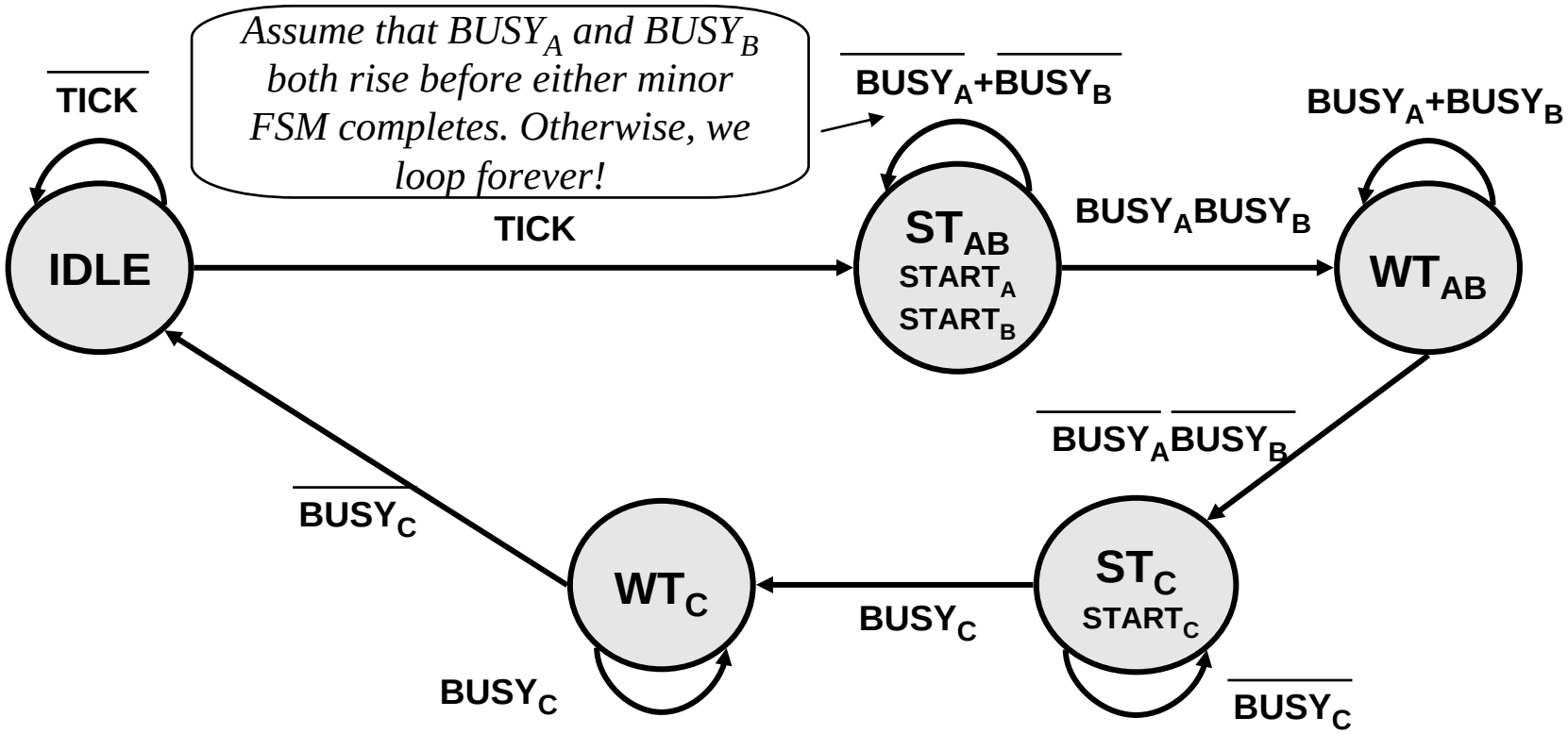


A Four-FSM Example



Operating Scenario:

- Major FSM is triggered by TICK
- Minors A and B are started simultaneously
- Minor C is started once both A and B complete
- TICKs arriving before the completion of C are ignored



Four-FSM Sample Waveform