Stopwatch implementation

Hardware Acceleration with FPGA

Speaker: Felipe Tortato

Agenda

- Presentation (10 min):
  - Solution Structuring
  - Simulation Results
  - Synthesis and Implementation
- Questions and discussion (10 min)

Solution Structuring

- Notation:
  - ‘1’ represents “active state” in the diagram
  - Transition of states happens with rising edge clock (AND specified signal)
  - If reset is active, current state will always the be starting one, independent on other inputs

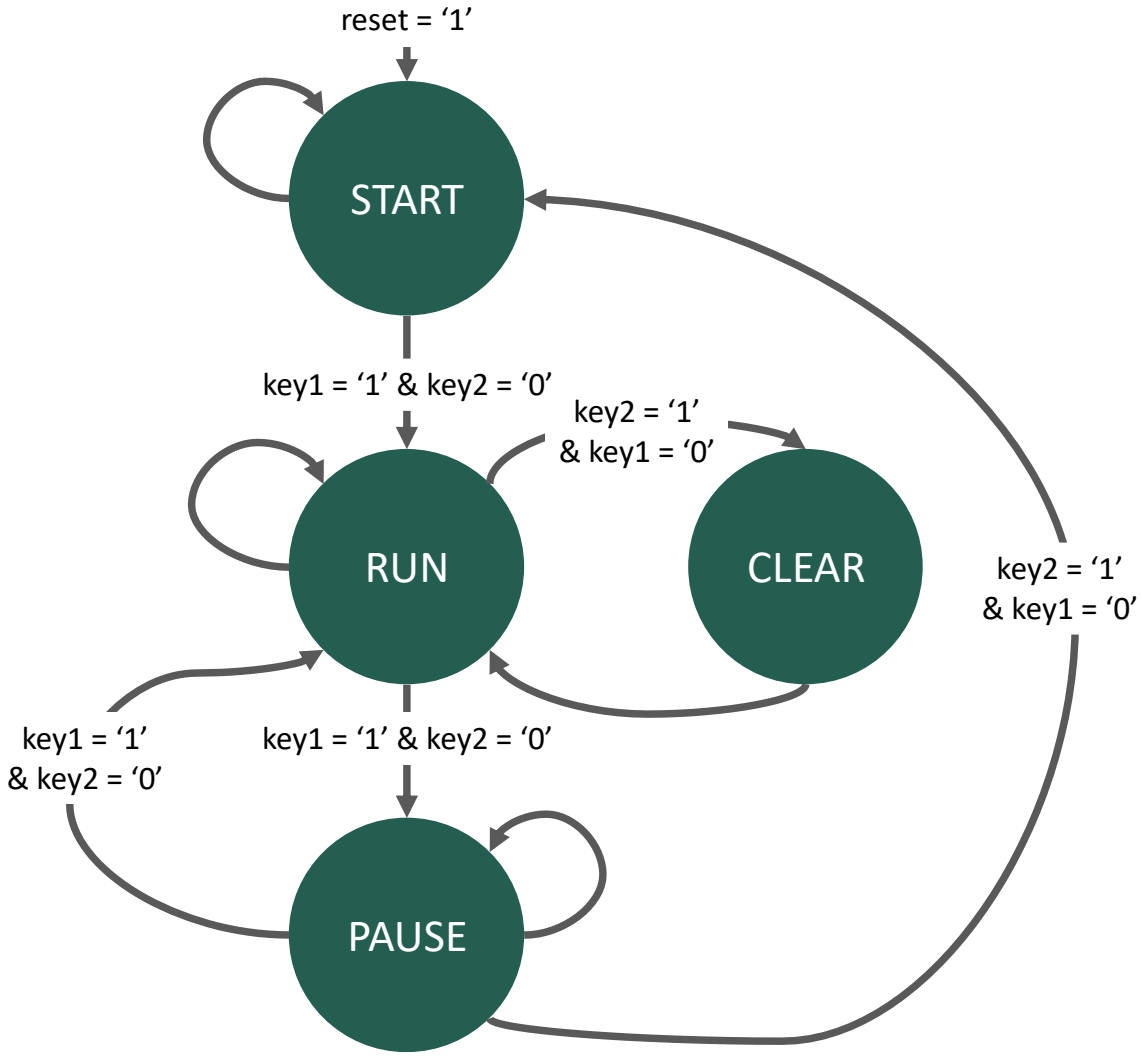


Figure 2: State diagram of the stopwatch (as example)

Solution Structuring

- Debouncer: Finite state machine with 3 states
  - START: beginning of execution, when button reset is pressed
  - IDLE: waiting for button press. The counter of cycles is cleared
  - WAITING: cycle counter is incremented every cycle while key is still pressed until reaching the limit. If key is released, it comes back to idle state. If the limit was achieved, a pulse of 1 cycle is set at the output, otherwise, the output remains at ‘0’

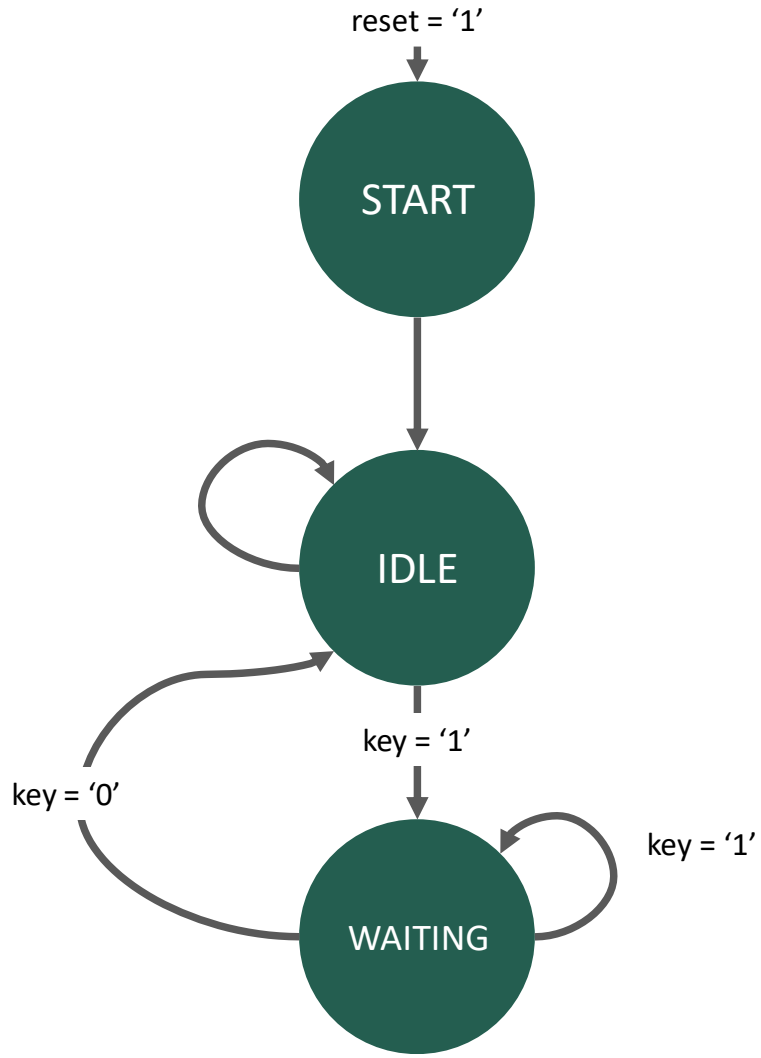


Figure 3: State diagram of the debouncer

Solution Structuring

- TDM: Finite state machine with 5 states
  - RESET: beginning of execution, when button reset is pressed. Counter of cycles is cleared
  - TD1: cycles are incremented until the limit is reached (time slot duration), then changes to TD2. The output is for the display 1
  - TD2: cycles are incremented until the limit is reached, then changes to TD3. The output is for the display 2
  - TD3: cycles are incremented until the limit is reached, then changes to TD4. The output is for the display 3
  - TD4: cycles are incremented until the limit is reached, then changes to TD1. The output is for the display 4

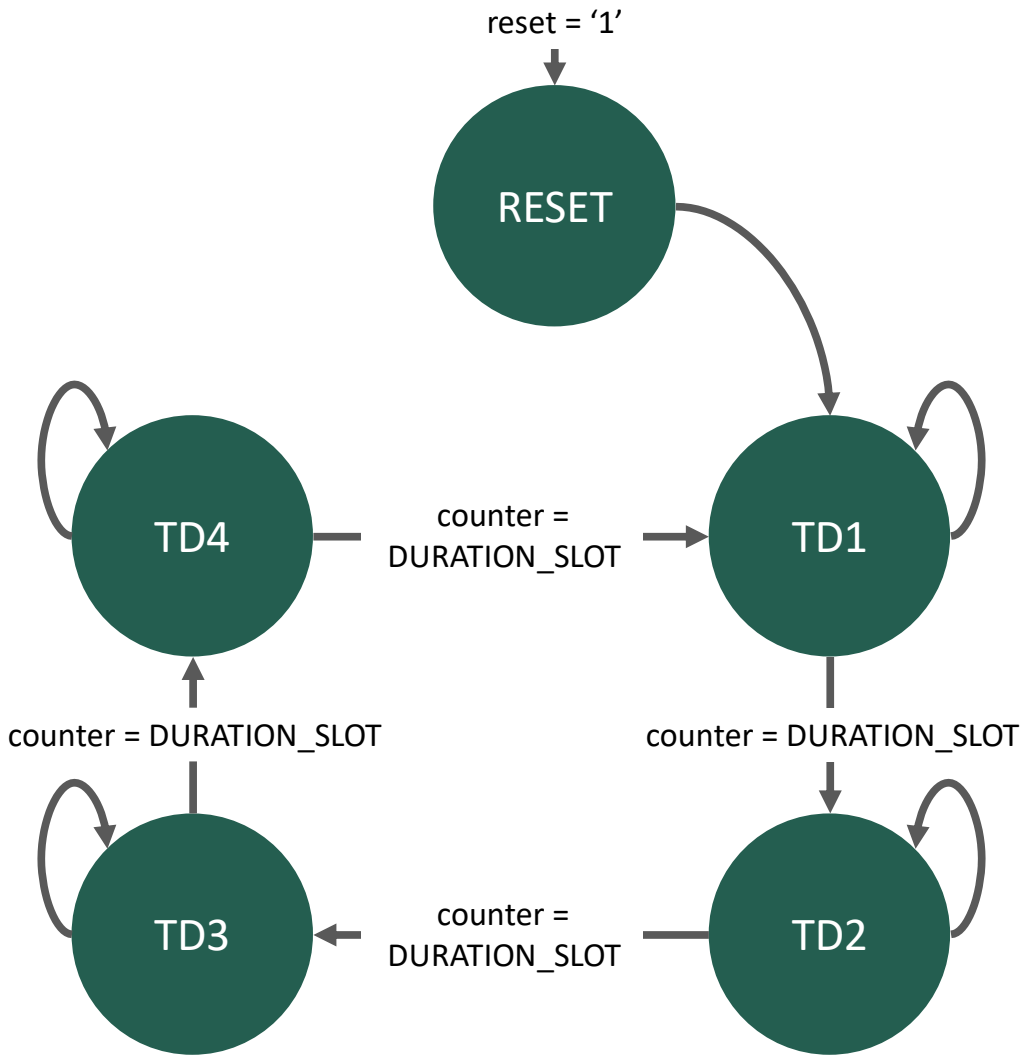


Figure 4: State diagram of the TDM

Solution Structuring

- SW: Finite state machine with 4 states
  - START: beginning of execution, when button reset is pressed. Cycles, seconds, tens of seconds, minutes and tens of minutes are cleared
  - RUN: cycles are incremented until a second is reached and then tens of seconds, minutes and tens of minutes are calculated
  - PAUSE: cycles are not incremented, but the value remains the same as the last cycle at RUN state
  - CLEAR: cycles, seconds, tens of seconds, minutes and tens of minutes are cleared. Comes back automatically to RUN state after one cycle

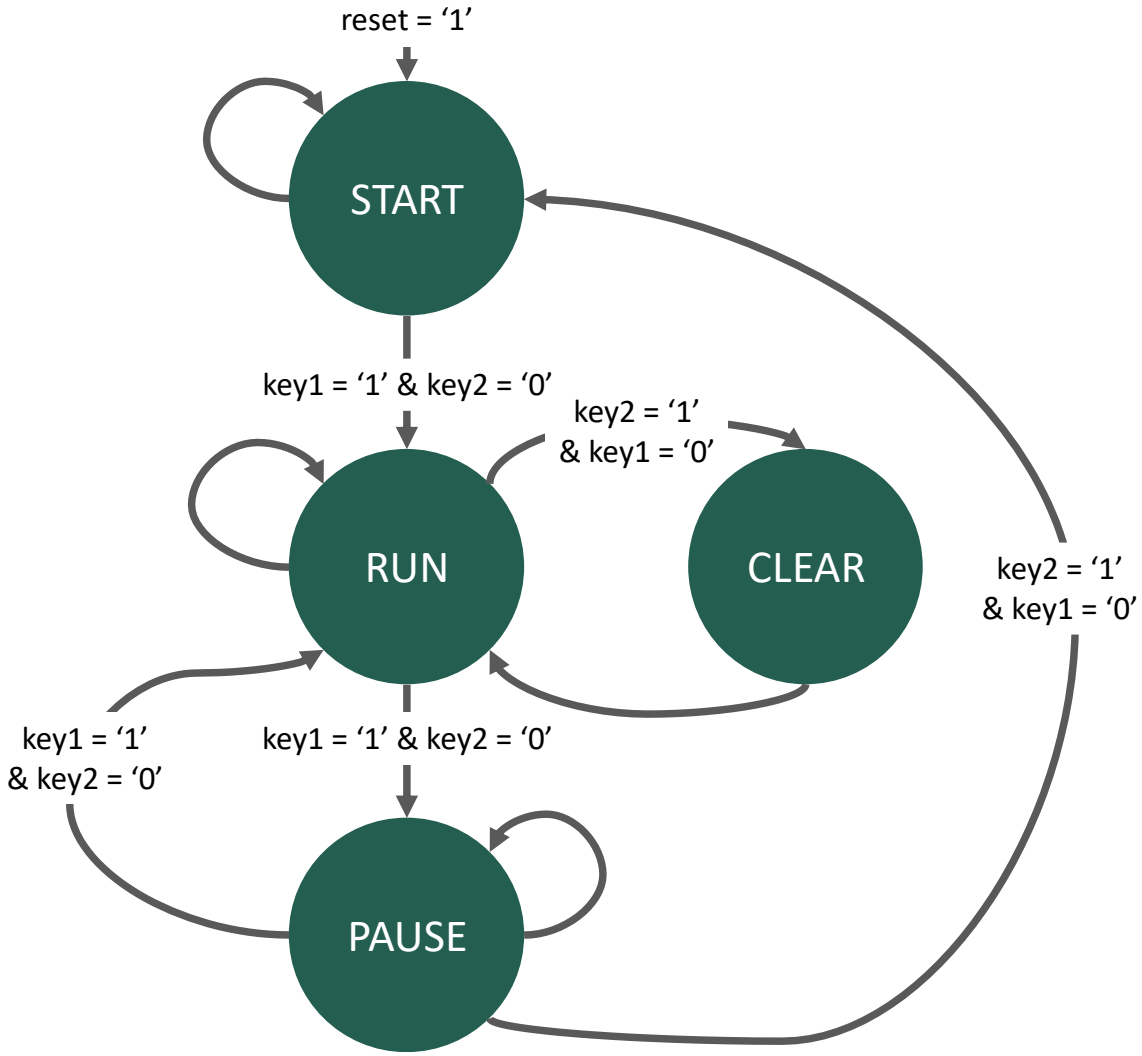


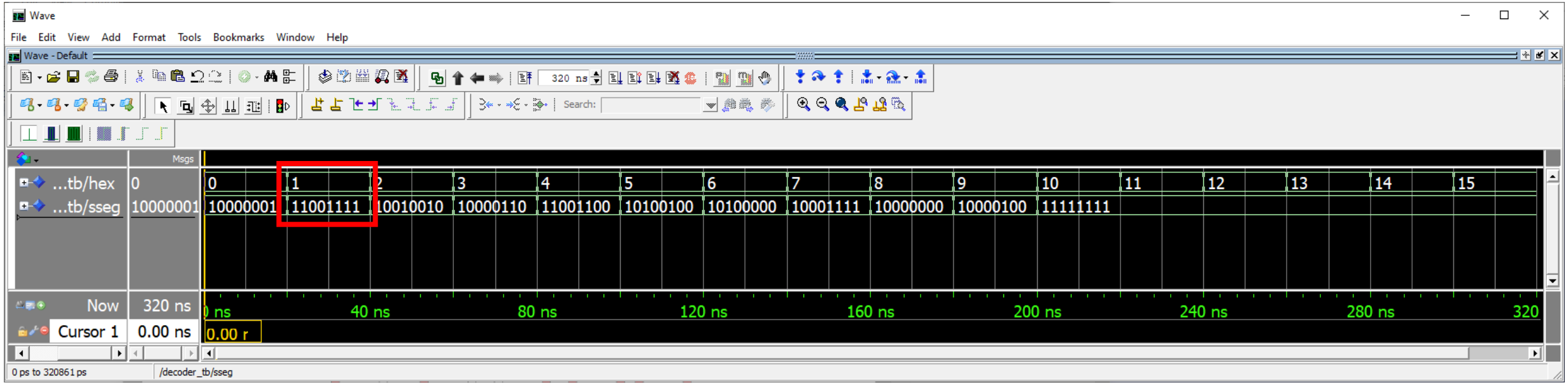
Figure 5: State diagram of the stopwatch

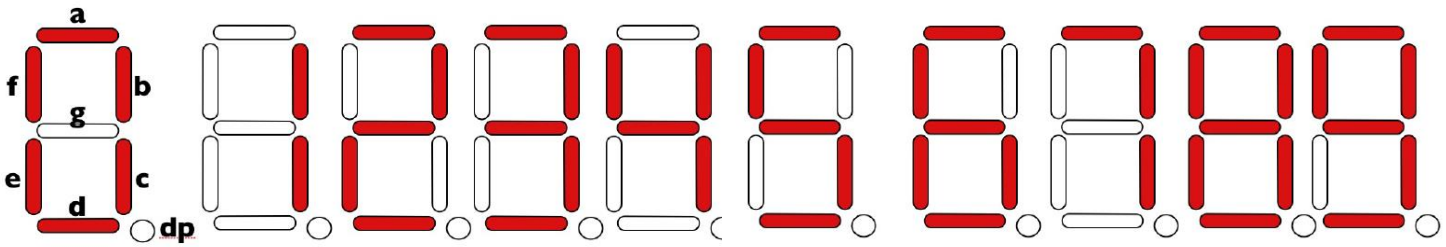
Simulation Results

- Structuring of testbench simulations
  - 7-segment display decoder (testbench given):
    - input of 4 bits is changed with numbers from 0x0 to 0xF -> the output is observed
  - Button debounce:
    - input is set to active for a smaller number of cycles than required -> the output is observed
    - input is set to active for the exact number of cycles required -> the output is observed
    - Input is set to active for a bigger number of cycles than required -> the output is observed
  - Time division multiplexer:
    - input is fixed with the numbers -> change in the output with time is observed
    - one of the input changes -> change in this time slot is observed
  - SW (Stopwatch controller + debounce + decoder):
    - variation of states using key1 and key2 -> displays are observed for a long number of cycles
- Use of generic was important to be able to easily change configuration of time for the test benches

Simulation Results

- 7-segment display Decoder ✓

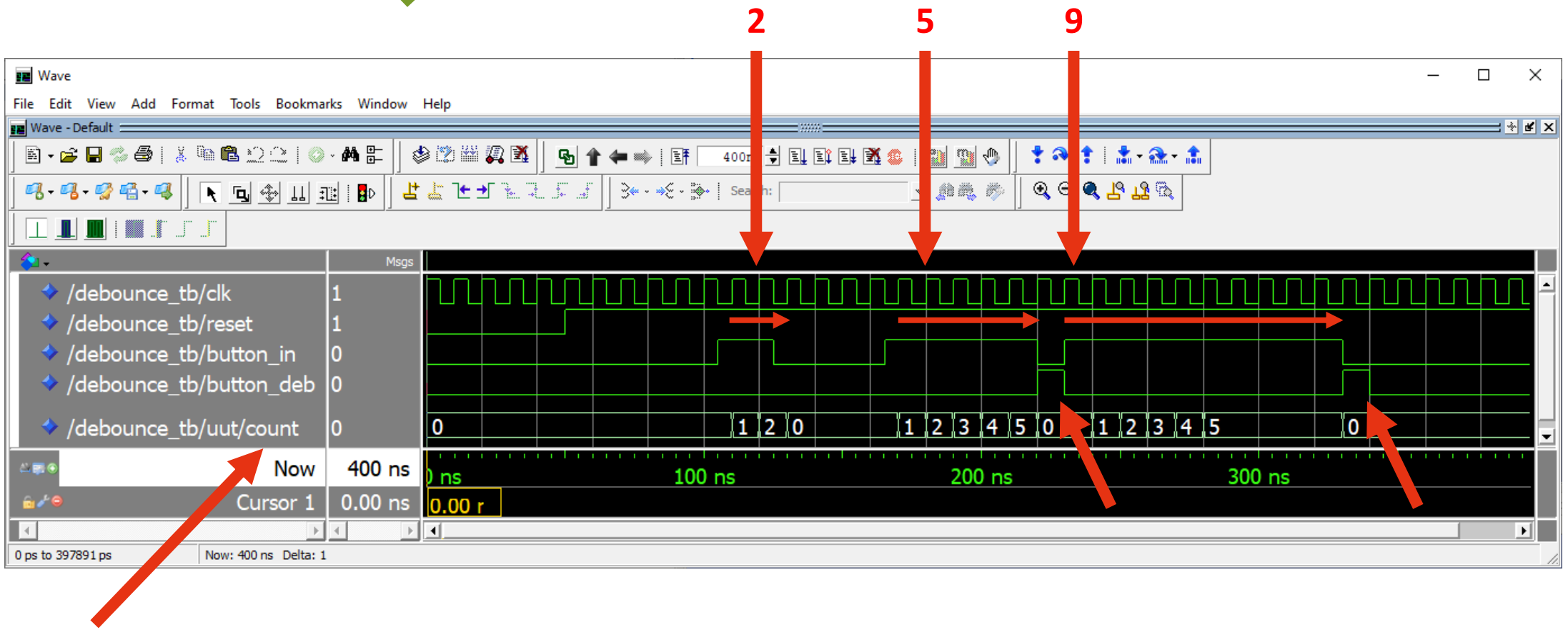




dp a b c d e f g
(7) ... (0)

Simulation Results

- Button Debounce ✓



Internal signal (debug only)

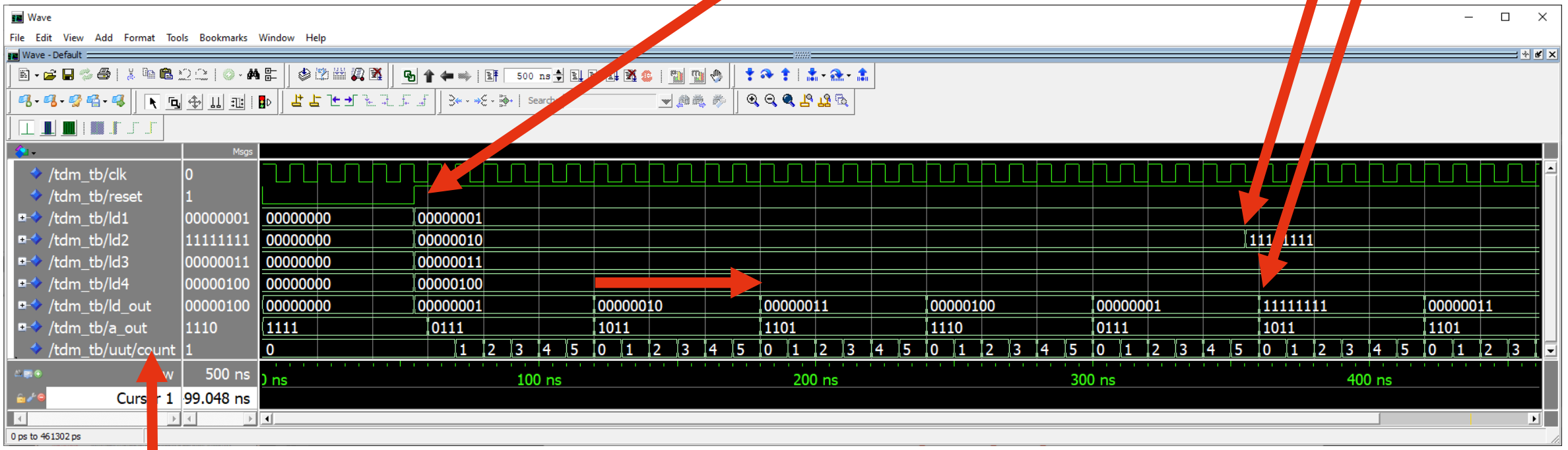
Button needs to be detected in active state for at least N (sim: 5) rising clock cycles, and need to stay one cycle at inactive state to produce the output

Simulation Results

- Time Division Multiplexer ✓

Reset release and start procedure

Change of a signal and change in its time window



Internal signal (debug only)

Simulation Results

- Stopwatch: key1 press, state change (start->run), counter increase

Key1 is released

Key1 is pressed

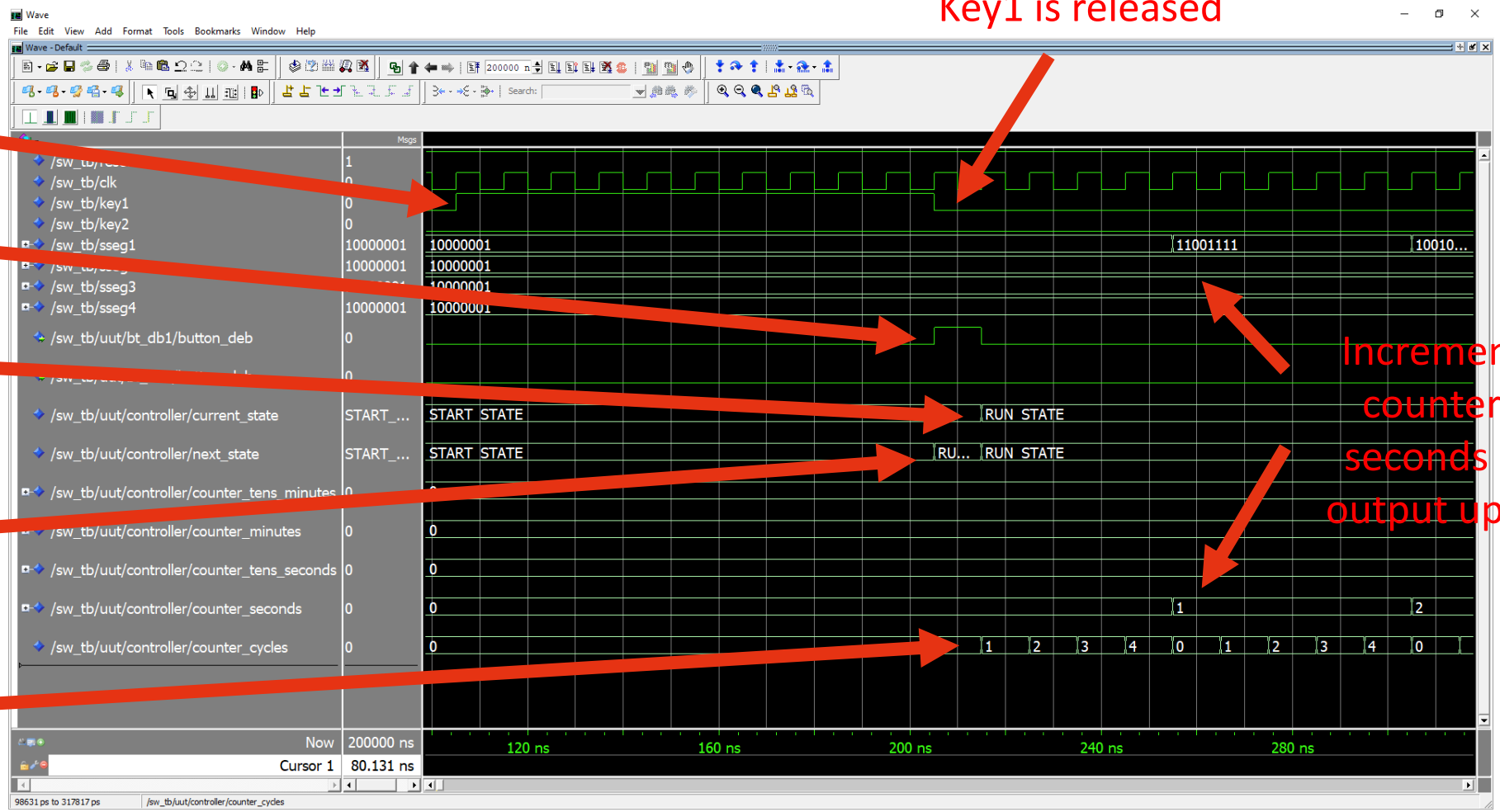
Debounce of key1 sends a pulse

Change of current state 1 cycle after

Change of next state with key1 press

Increment in counter of cycles

Increment in counter of seconds and output update



Synthesis

Summary per component:

	Decoder	Debounce	TDM	Stopwatch
States	-	3	5	4
Transitions	-	7	9	11
D-Type FFs	-	2	1	42
Adder/Subtractor	-	-	-	5
Comparators	-	-	1	4
Counters	-	1	1	-
ROMs	1	-	-	-
State Encoding	-	gray	one hot	gray

Synthesis

Macro statistics:

Counters of controller interpreted as registers

	Total	Used at
Registers	**7**	
1-bit	2	Stopwatch controller and TDM
4-bit	4	Stopwatch controller
25-bit	1	Stopwatch controller
Adders/Subtractors	**5**	
4-bit adder	4	Stopwatch controller
25-bit adder	1	Stopwatch controller
ROMs	**4**	
16x8-bit	4	Decoder
Counters	**3**	
5-bit up	2	Debounce
17-bit up	1	TDM
Comparators	**5**	
4-bit less	4	Stopwatch controller
17-bit less	1	TDM

Synthesis

Device utilization (map report):

- Number of Slices: 132/3584 (3%)
- Number of Slice Flip Flops: 100/7168 (1%)
- Number of 4 input LUTs: 233/7168 (3%) – logic: 176; route-thru: 57
- Number of IOs: 16
- Number of bonded IOBs: 16/97 (16%)
- Number of GCLKs: 1/8

Synthesis

Timing report:

- Minimum period: 7,751 ns
  - controller/current_state_FSM_FFd2 to controller/counter_tens_minutes_1
  - controller/current_state_FSM_FFd2 to controller/counter_tens_minutes_2
- Maximum allowed clock frequency: 129,016 MHz
- Worst case slack:
  - Setup: 42,249 ns
  - Hold: 0,757 ns
    - time_multiplexer/current_state_FSM_FFd4 to time_multiplexer/current_state_FSM_FFd2

Conclusion and suggestions for future work

- Solution worked
  - Adjustments in the debounce time and in the order of the displays were required before having a full working version
- Better structuring of the solution before start of development
  - Need of TDM only noticed after almost everything was ready
  - Optimization in state machines could lead to a smaller number of bits required for representation
  - Lack of experience led to adaptations to get it working
  - TDM before the decoder could have used less hardware
- Automatization of the test procedure
  - Checking outputs in the test bench, for validation without need of visual check

Thank you!

Hardware Acceleration with FPGA

Speaker: Felipe de Souza Tortato