

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ І СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА КОНСТРУЮВАННЯ КЕОА

Розрахунково – графічна робота

по курсу

«Обчислювальні та мікропроцесорні засоби в радіоелектронній апаратурі — 1»

за темою

“Watchdog timer”

Виконав:

студент гр. ДК-61

Шваюк М.В.

Перевірив:

Тимофій Ходнєв

Київ-2018

Вступ

Моїм завданням на РГР було створення модуля watchdog — timer до мого software процесору, що базується на архітектурі MIPS. Watchdog timer є абсолютно необхідним елементом будь-якої Embedded системи, а особливо тих, що мають працювати без людського втручання впродовж тривалого часу.

Нехай у нас є Embedded система, яка працює віддалено, без втручання людини. Це може бути метеорологічний датчик, розташований високо в горах, датчик вологості ґрунту на полі, на основі чіїх даних система приймає рішення про те чи варто починати полив, Wi-fi маршрутизатор, який стоїть десь в коморі і до якого людина не має ніколи підходити, або навіть марсохід, доставка якого на іншу планету коштувала сотні мільйонів долларів та сотні тисяч людино-годин. Тепер давайте уявимо, що всередині керуючого процесора цих пристроїв стався збій на рівні заліза і він просто “завис”, перестав приймати команди і відповідати на них. У такому випадку систему необхідно перезавантажити, щоб вивести її з цієї мертвої точки. У випадку Wi-fi роутера або датчика це зробить людина. Якщо ж таке сталося в метеодатчику, який знаходиться за десятки кілометрів від цивілізації, або тим паче марсоході, до якого фізично неможливо дістатися то наслідком цього будуть великі фінансові збитки. Також не можна забувати про медичні системи, або системи забезпечення безпеки у авіалайнерах, потягах, автомобілях, від яких залежать людські життя і неприпустимо щоб там могла бути хоча б мінімальна вірогідність виходу приладу з ладу і “зависання”. Для розв'язання цієї проблеми і був придуманий watchdog timer.

Опис роботи watchdog timer і їх види

Весь сенс роботи watchdog timer полягає у наступному: він має постійно періодично отримувати сигнал від процесора про те, що система функціонує справно та не зависла. Будемо називати інструкцію, яка генерує такий сигнал **CLRWDT**. Якщо протягом певного періоду часу такий сигнал не приходить — відбувається зкидання системи.

Моя реалізація watchdog timer буде працювати наступним чином:

- Впродовж часу ***t_wait*** watchdog очікує на сигнал **CLRWDT**. Якщо його не було — відбувається перехід у стан ***safe_state***. У цьому стані система “завмирає” (двигуни перестають рухатися, датчики — записувати дані, передавачі — передавати інформацію тощо).
- Watchdog очікує ще один період часу ***t_wait*** щоб дати процесору можливість вийти із зависання без перезавантаження. Якщо в цей проміжок часу відбувається приходить сигнал **CLRWDT** — система повертається до нормальної роботи, у іншому випадку відбувається ***hard_RESET*** — hardware перезавантаження системи, яке досягається утриманням активного логічного рівня на виводі RESET мікроконтролера.
- Watchdog утримує активний логічний рівень на виводі RESET впродовж часу ***t_reset***, потім система перезавантажується та повертається до нормальної роботи.

Тривалість проміжків часу ***t_wait*** та ***t_reset*** можна регулювати, записуючи їх значення у відповідні регістри.

Сигнал **CLRWDT** може приходити асинхронно, система це передбачає.

Очевидно, що в основі watchdog timer має лежати лічильник, що рахує вниз, який при надходженні сигналу **CLRWDT** буде зкидатися і в нього буде записуватися значення, яке визначає ***t_wait***.

В моїй реалізації використано 3 лічильника:

- **counter_1** — коли значення у ньому стане рівне нулю, запуститься режим ***safe_state***;
- **counter_2** — коли значення у ньому стане рівне нулю, запуститься режим ***hard_RESET*** — на виводі RESET мікроконтролера встановиться значення активного логічного рівня;
- **counter_3** — коли значення у ньому стане рівне нулю, режим ***hard_RESET*** завершиться, на виводі RESET мікроконтролера встановиться неактивний логічний рівень, і система почне працювати з початку.

Вихідний код на мові Verilog

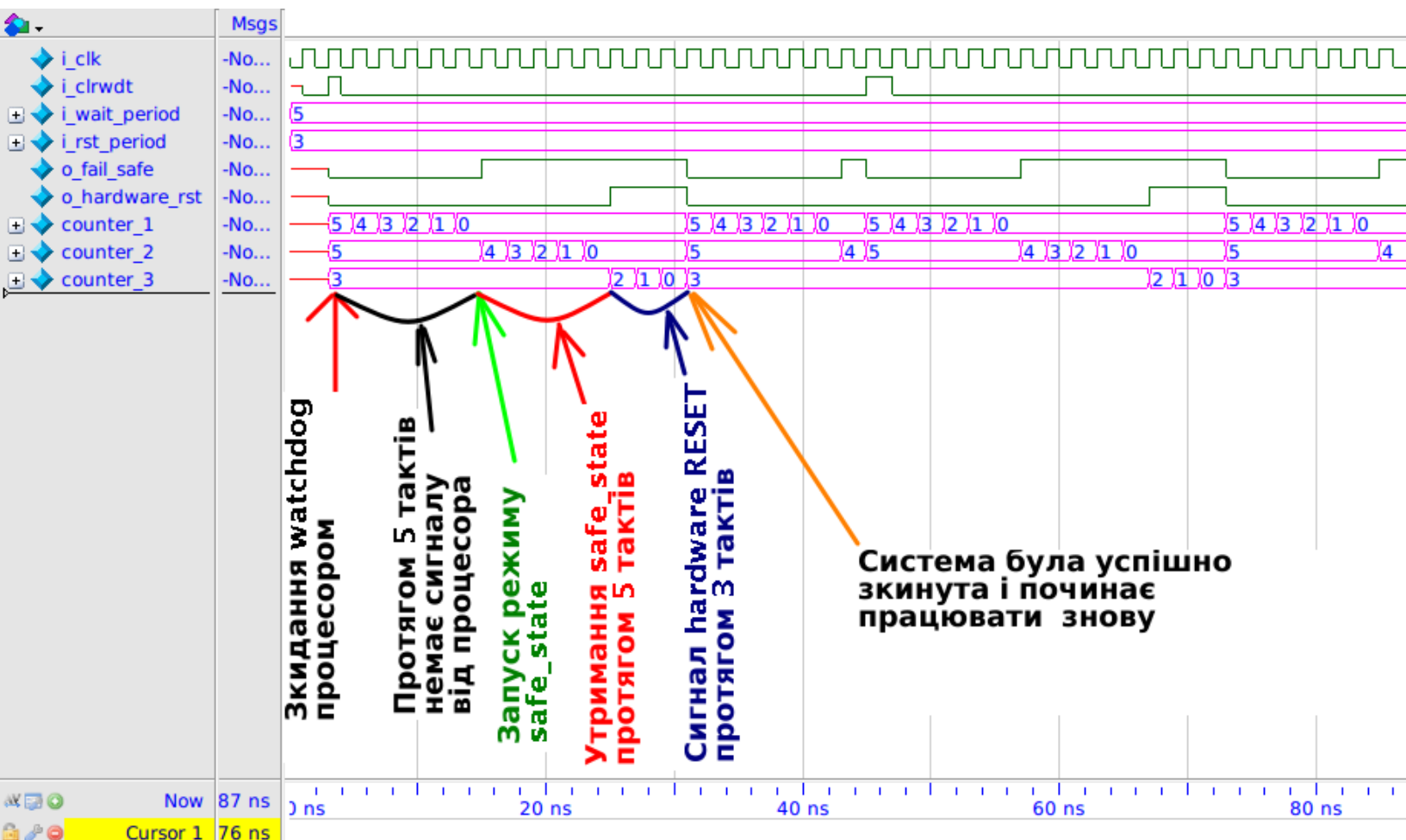
З вихідним кодом ви можете ознайомитися за посиланням

https://github.com/maximwowpro/Max-Shvayuk-comp_arch-and-MIPS-labs/tree/master/RGR_watchdog

Симуляція у ModelSim

Для перевірки роботи схеми, запустимо мій watchdog у симуляторі. Тестбенч можна знайти у папці з вихідним кодом.

Отримали такі результати:

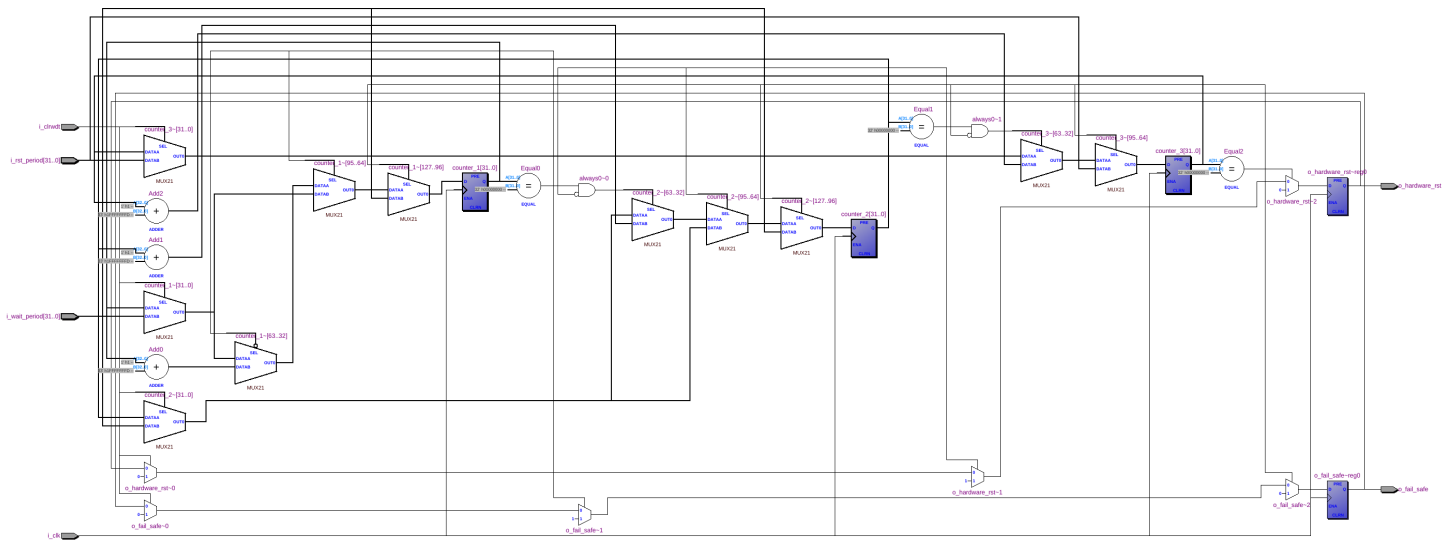


На даній waveform:

- `i_clk` — тактова частота watchdog timer ;
- `i_clrwdt` — сигнал зкидання таймера (має надсилатся процесором) ;
- `i_wait_period` — проміжок часу t_{wait} ;
- **`i_rst_period`** — проміжок часу t_{reset} ;
- `o_fail_safe` — сигнал, високий рівень якого переводить систему в стан ***safe_state***, коли вона “завмирає” ;
- `o_hardware_rst` — сигнал ***hard_RESET***, його тривалість визначається значенням `i_rst_period` ;
- `counter_1`, `counter_2`, `counter_3` — виводять значення трьох внутрішніх лічильників watchdog timer (додано лише для наочності).

Синтез у FPGA

Я проводив синтез під FPGA Intel DE-2 EP2C35F672C6. Мій watchdog timer синтезувався у наступну схему:



Вона займає на платі стільки ресурсів:

Total logic elements	140 / 33,216 (< 1 %)
Total combinational functions	137 / 33,216 (< 1 %)
Dedicated logic registers	98 / 33,216 (< 1 %)

Для будь — якого сучасного FPGA це абсолютно невелика цифра, яку можна спокійно виділити під такий важливий та корисний модуль.

Я, звісно, розумію, що та схема, у яку компілятор синтезував мій код не оптимальна і можна створити її набагато ефективніше, якщо промалювати схему собі на папірці, потім самому створити модулі лічильників на Т-тригерах,

поєднати їх за допомогою добре продуманої комбінаційної логіки і отримати оптимальну реалізацію watchdog timer, але такий підхід вимагає багато часу на розробку, тому під час проектування сучасних систем FPGA або ASIC використовується рідко, зазвичай коли необхідно отримати максимум функціоналу з якогось не дуже потужного чіпа.

Висновок

У ході РГР мною було досліджено та власноруч реалізовано таку цифрову схему як watchdog timer. Це незамінний елемент будь-якого сучасного мікропроцесора. Також я приєднав watchdog timer до власного 1-тактного процесора на основі архітектури MIPS, чим розширив його функціонал. Також я переконався у життєздатності цього рішення, створивши тестову програму для процесора.