Trigger – opis modułu

Mateusz Danilczuk

1. Porty I/O

Nazwa	Kierunek	Rozmiar	Funkcja
Clk320	IN	1	Zegar
Mt_cou	IN	[2:0]	Określa jakie operacje ma wykonywać trigger
CH_trigt	IN	[11:0]	Określa które kanały dla czasu są aktywne
CH_triga	IN	[11:0]	Określa które kanały dla ładunku od strony A są aktywne
CH_trigb	IN	[11:0]	Określa które kanały dla ładunku od strony C są aktywne
CH_TIME_T	IN	(0:11)[9:0]	Czas z poszczególnych kanałów
CH_ampl0	IN	(0:11)[12:0]	Ładunek z poszczególnych kanałów
Tcm_req	IN	1	Dodatkowy sygnał służący do sterowania działaniem triggera
tt	OUT	[1:0]	Semi-central trigger, central trigger
ta	OUT	[1:0]	OR A trigger, OR C trigger

2. Sygnały wewnętrzne

Nazwy	Rozmiar
TT_mode, Trig_onA0, Trig_on0, req_act, req_done, Trigb0, Bsw	1
CH_ampl	(0:11)[12:0]
CH_TIME1_T	(0:11)[9:0]
N_chans	[3:0]
tts, tas	[5:0]
ta0s	[6:0]
tts_c, tas_c	[3:0]
tas00, tas01, tas10, tas11, tai00, tai01, tai10, tai11, tai20, tai21, tts00, tts01, tts10, tts11	[2:0]
N1_chans, N2_chans	[2:0]

3. Działanie triggera

Trigger działa na zboczu narastającym zegara. Najpierw sprawdzany jest stan wejścia ***tcm_req***. W przypadku stanu wysokiego, rejestr ***req_act*** jest również ustawiany na stan wysoki. W przypadku gdy *tcm_req* nie równa się „1" sprawdzany jest warunek ***req_done = '1' and mt_cou = „010"***. Jeśli jest on prawdziwy, ***req_act*** ustawiany jest na zero. ***Req_act*** jest później wykorzystany jako warunek w jednym przypisaniu sygnału.

Następnie następuje sprawdzenie wartości na wejściu ***mt_cou[2:0]***.

Jeśli *mt_cou* równa się „000" rejestr *Bsw* ustawiany jest na „1", w innym wypadku na „0". Rejestr ten zostanie później wykorzystany do określenia wartości ***tt.***

Jeśli ***mt_cou*** równa się „001" rejestr ***TT_mode*** ustawiany jest na „1", w innym wypadku na „0". Potem obliczana jest operacja logiczna OR na wszystkich bitach sygnałów wejściowych ***CH_triga*** i ***CH_trigb***. Wynik tej operacji zapisywany jest do rejestrów ***Trig_onA0*** i ***Trigb0***. Oprócz tego do rejestrów ***tai00*** i ***tai01*** przypisywane są sumy wartości najmniej znaczących bitów z każdego kanału, przy czym ***tai00*** obejmuje kanały od 0 do 5 a ***tai01*** od 6 do 11. Analogiczna operacja wykonywana jest na drugich i trzecich najmniej znaczących bitach i jest zapisywana do rejestrów ***tai10***, ***tai11***, ***tai20***, ***tai21***.
Wartości z tych rejestrów są później sumowane ze sobą i zapisywane do rejestru ***ta0s***, przy czym bity są przesunięte o jeden w lewo dla ***tai10***, ***tai11*** oraz o dwa dla ***tai20***, ***tai21***. W rezultacie sumy drugich i trzecich najmniej znaczących bitów z posczególnych kanałów liczą się odpowiednio dwa oraz cztery razy bardziej do ostatecznego wyniku.

Następnie ***mt_cou*** jest porównywane do „010". W przypadku gdy nie jest równe „010" następuje seria przypisań tj.:

> Dla każdego kanału od 0 do 11:
> ***CH_TIME1_T[9:0] <= CH_TIME1_T[9]&CH_TIME1_T[9]&CH_TIME1_T[9:2]***
> ***CH_ampl[12:0] <= CH_ampl[9]&CH_ampl[9]&CH_ampl[9:2]***
>
> Ponadto:
> ***tts_c <= tts[5:2], tas_c <= tas[5:2]***

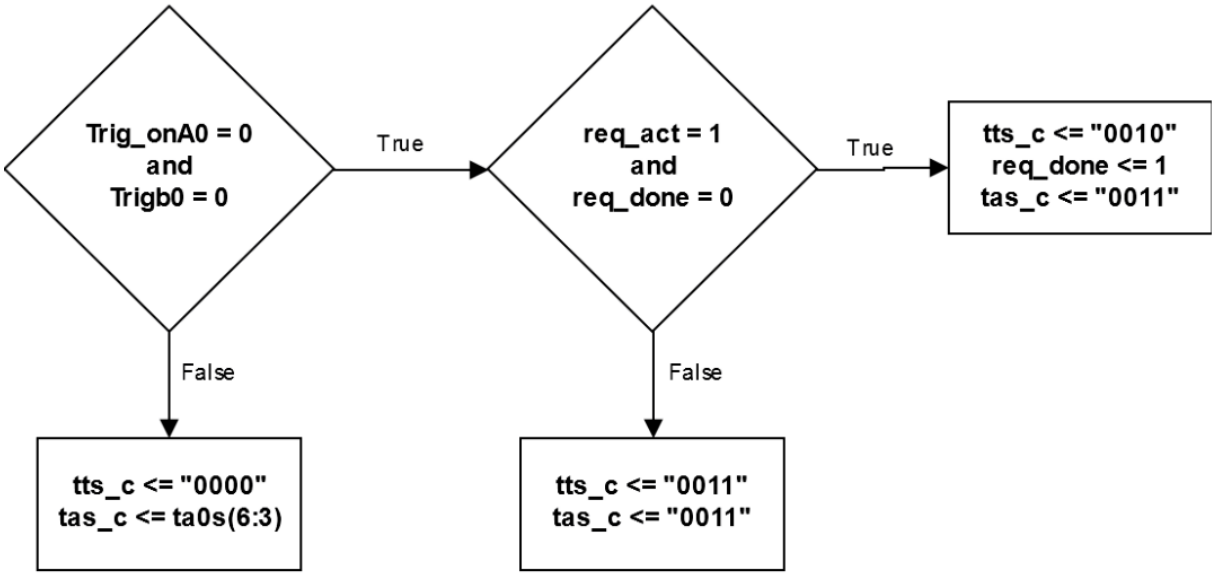
Są to zmienne pomocnicze użyże później przy innych operacjach.

Jeśli ***mt_cou*** jest równe „010” następuje przypisanie, po którym odbywa się zagnieżdżone sprawdzanie warunków. Przypisanie to wygląda następująco:

Dla każdego kanału od 0 do 11:
CH_TIME1_T <= ***CH_TIME_T***
CH_ampl <= ***CH_ampl0***

Dodatkowo jeśli rejestr ***req_done*** wynosi „1”, przypisywana jest mu wartość „0”.

Następne przypisania najłatwiej zobrazować w postaci diagramu (Rys.1). W zależności od wartości rejestrów ***Trig_onA0, Trigb0, req_act i req_done***, wartości rejestrów ***tts_c*** i ***tas_c*** przyjmują różne wartości.



Rys.1. Algorytm określania wartości tts_c i tas_c

Po zakończeniu sprawdzania wartości ***mt_cou*** następuje seria przypisań i sumowań.

Wpierw zostaje obliczona wartość ***Trig_on0*** (analogicznie do ***Trig_onA0*** i ***Trigb0*** jest to operacja OR wykonana na wszystkich bitach ***CH_trigt***).
Następnie do rejestrów ***N1_chans*** i ***N2_chans*** przypisywana jest suma bitów z ***CH_trigt*** (kanały 0 do 5 dla ***N1_chans*** oraz 6 do 11 dla ***N2_chans***).
Potem zostaje sprawdzony warunek ***not Trig_on0 and (Trig_onA0 or Trigb0)*** (jest on prawdziwy wtedy, gdy wszystkie kanały ***CH_trigt*** są nieaktywne oraz przynajmniej jeden kanał ***CH_triga*** lub ***CH_trigb*** jest aktywny). W przypadku, gdy jest on prawdziwy do rejestru ***N_chans*** zostaje przypisana suma ***N1_chans*** i ***N2_chans***, w innym wypadku przypisane

zostaje „1101” (13 binarnie).
Następnie sumowane są ze sobą najmniej znaczące i drugie najmniej znaczące bity ***CH_TIME1_T*** i ***CH_ampl***. Analogicznie jak w przypadku ***ta0s*** otrzymane sumy są ponownie sumowane ze sobą z uwzględnieniem przesunięcia bitowego w lewo. Dodatkowo do sumy zostają dołączone wartości rejestrów odpowiednio ***tts_c*** oraz ***tas_c***.

Po tych operacjach dochodzi do ustalenia wartości wyjść ***tt*** i ***ta***. Do ich określenia zostają użyte wartości w rejestrach stanu ***TT_mode***, ***Bsw*** (ustalone wcześniej podczas sprawdzania wartości ***mt_cou***).
Dochodzi wówczas do następujących przypisań w zależności od spełnionych warunków:

```
if (TT_mode = „1”)
        tt <= N_chans[1:0]
else if (Bsw = „1”)
        tt <= Trigb0 & tts[0]
else
        tt <= tts[1:0]


if (TT_mode = „1”)
        ta <= N_chans[3:2]
else
        ta <= tas[1:0]
```

4. Komentarz końcowy

Kod trigger.vhd generalnie przypomina opis modułu zawarty w postaci schematów blokowych, jednak spora część kodu jest trudna do zrozumienia. Brakuje także kilku operacji, mianowicie nie ma nigdzie zawartego uśredniania, ani odejmowania czasów pomiędzy zdarzeniami. Ponadto nie zgadza się liczba portów wyjściowych (brak Vertex trigger). Być może brakująca funkcjonalność znajduje się w układzie ASIC, który również został użyty w projekcie.