Podczas analizy wyników testów należy brać pod uwagę, że rdzeń Cortex-M0 zastosowany w mikrokontrolerach LPC1114 charakteryzuje się mniejszą wydajnością niż Cortex-M3 zastosowany mikrokontrolerach STM32 lub LPC1700. W ofercie NXP znajduje się mikrokontroler LPC1769, który katalogowo może być taktowany sygnałem zegarowym o częstotliwości do 120 MHz. Jego także poddamy testom redakcyjnym i przedstawimy Czytelnikom wyniki naszych badań.

Podkręcanie zegarów

Overclocking w mikrokontrolerach

Sezon urlopowy w pełni, lutownice zamieniliśmy na leżaki, a Elektronikę Praktyczną, zamiast przy stole warsztatowym, czytamy na świeżym powietrzu. Jak zwykle w „sezonie ogórkowym" publikujemy artykuły o nieco lżejszym charakterze i taki również będzie artykuł o tematyce overclocking'u.

Tematyka „podkręcania zegarów" jest głównie związana z komputerami PC. Zachęcają do tego niejednokrotnie sami producenci podzespołów komputerowych. Każda dobra płyta główna ma w BIOS-ie pozycję menu M.I.T. Inteligent Tweaker, umożliwiającą ustawianie parametrów różnych podzespołów komputera na wartości ponadnormatywne. Istnieją kluby związane z tą tematyką, ze współzawodnictwem polegającym na uzyskaniu jak największej częstotliwości taktowania. Przy częstotliwościach, z jakimi pracują współczesne komputery, jedną z najistotniejszych kwestii jest chłodzenie. W ekstremalnych przypadkach stosuje się chłodzenie ciekłym azotem, uzyskując częstotliwości taktowania rzędu 6 GHz. W czeluściach Internetu natknąłem się również na bardzo kontrowersyjny sposób chłodzenia polegający na umieszczeniu całego peceta włącznie z zasilaczem w akwarium wypełnionym olejem transformatorowym (**fotografia 1**). Zastanawiam się tylko, jak na taki wynalazek zareagują tworzywa sztuczne stosowane na płycie głównej oraz złącza pomiędzy modułami, które raczej nie są odporne na działanie oleju. Ciekawi mnie również wymiana podzespołów w takim komputerze.

Czy obserwując dość interesujące rezultaty overclockingu komputerów PC, nie zastanawialiście się czasem czy można też „podkręcić" mikrokontroler?

Częstotliwości pracy mikrokontrolerów są o rzędy wielkości mniejsze, więc w naszych testach nie będziemy musieli uciekać się do ekstremalnych sposobów chłodzenia, a wszystkie testy będziemy przeprowadzać w warunkach standardowych, przy chłodzeniu naturalnym oraz nominalnym napięciu zasilania. Na warsztat postanowiliśmy wziąć cztery mikrokontrolery najpopularniejszych rodzin: ATmega32 – przedstawiciel rodziny AVR, LPC2142 – przedstawiciel rodziny ARM7TDMI, STM32F107VBT6 – przedstawiciel popularnej rodziny Cortex-M3 oraz ostatni krzyk mody, który może się okazać potencjalnym zabójcą popularnych AVR-ów, przedstawiciel rodziny ARM Cortex-M0 – LPC1114. Przy okazji testów overclockingu zrobimy testy wydajności jednostek centralnych oraz sprawdzimy, jaka jest w rzeczywistości deklarowana przez producentów wydajność obliczeniowa jednostek.

Założenie, algorytm testowy, uwagi dotyczące działania układów peryferyjnych

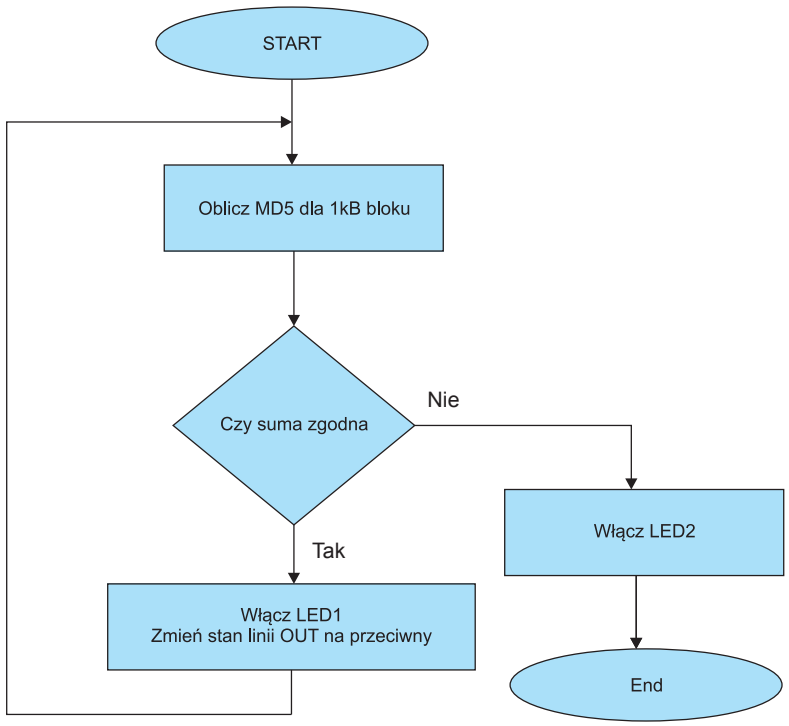
Celem testu jest zbadanie podatności na overclocking jednostki CPU w wybranych

Fotografia 1. Dzieło firmy Puget Systems – PeCet zanurzony w chłodzącym go oleju transformatorowym

mikrokontrolerach. Należy podkreślić, że prawidłowe działanie jednostki centralnej przy określonej częstotliwości sygnału zegarowego nie gwarantuje, że pozostałe układy peryferyjne mikrokontrolera będą działały prawidłowo. Badanie układów peryferyjnych pod względem prawidłowości działania przy podwyższonej częstotliwości wykracza poza łamy niniejszego artykułu i powinno być przeprowadzone oddzielnie. Bardziej zaawansowane, 32-bitowe mikrokontrolery najczęściej mają oddzielny dzielnik częstotliwości, umożliwiający taktowanie poszczególnych układów peryferyjnych niezależnie. Istnieje zatem możliwość zmniejszenia częstotliwości pracy tych układów do wartości nieprzekraczających częstotliwości nominalnych, co powinno gwarantować ich poprawną pracę. W przypadku prostych układów 8-bitowych, możliwości takiej najczęściej nie mamy i układy peryferyjne taktowane są z taką samą częstotliwością, jak rdzeń mikrokontrolera. Ponieważ przy przetaktowaniu jednostka centralna może pracować nie w pełni poprawnie, np. dokonując błędnych obliczeń, jako algorytm testowy wybrano bardziej zaawansowany test, od zmiany stanu diod LED w pętli. Polega on na wykonywaniu cyklicznie algorytmu MD5 (ramka) dla 1 kB bloku danych umieszczonego w pamięci Flash oraz porównaniu wyników obliczeń z wynikiem wzorcowym (**rysunek 2**).

Gdy wyznaczona suma MD5 jest zgodna, wówczas linia GPIO oznaczona jako OUT zmieniana jest na stan przeciwny, równocześnie włączana jest dioda LED D1, a następnie cykl powtarza się od początku. W przypadku, gdy wyznaczona suma MD5 nie jest prawidłowa, wówczas zapalana jest dioda LED2 oraz wykonywana jest pętla nieskończona. Zastosowanie powyższego algorytmu pozwala mieć minimum pewności co do prawidłowej pracy rdzenia, oraz pamięci Flash przy wybranej częstotliwości. Mierząc okres sygnału na wyjściu OUT możemy również obliczyć czas wykonywania wyliczania sumy MD5 z 1 kB bloku danych. Aplikacja testowa komunikuje się z otoczeniem tylko za pomocą linii GPIO, dzięki czemu wyeliminowano potencjalne problemy z układami peryferyjnymi.

Aplikacja testowa została zaprojektowana za pomocą GNU Make z użyciem kompilatora gcc (dla architektury AVR: avr-gcc, dla architektury AMD64: gcc, dla architektury ARM: arm-none-eabi-gcc). Składa się ona z części wspólnej zawierającej bibliotekę obsługi MD5 oraz funkcję wyznaczającą MD5 z 1 kB bloku danych wejściowych. Interfejs użytkownika zawiera tylko jedną funkcję do_test(), która w przypadku, gdy suma została wyznaczona prawidłowo zwraca false, natomiast w przypadku błędu zwraca wartość true. Biblioteka wyznaczająca MD5 została pobrana z SourceForge: *http://sourceforge.net/projects/md5-utils/*. W kodzie wprowadzono jedynie drobne modyfikacje dla 8-bitowej architektury AVR. W katalogach atmega32, lpc2142, stm32f107, x86, lpc1111 znajduje się kod specyficzny dla poszczególnych uC oraz dla komputera PC. Czas wykonywania algorytmu MD5 na komputerze PC jest czasem wzorcowym, w odniesieniu do którego będzie porównana wydajność poszczególnych mikrokontrolerów. Kod dla komputera PC przeznaczony jest dla Linux-a i został skompilowany dla architektury AMD64 (kod 64-bitowy). Pomiar czasu wykonania odbywa się poprzez 1000-krotne wywołanie funkcji do_test(), a następnie wyznaczenie średniego czasu jej wykonania. Kod 64-bitowy został wykonany na procesorze Phenom II, dla którego wyznaczanie 1 kB bloku zajmuje 2,675 µs, co po przeliczeniu daje czas wykonania rzędu 7,49 ms/MHz. Będzie to wzorcowy czas wykonania testu. Kompilacja programu testowego odbywa się z wiersza polecenia. Aby skompilować program, należy wejść do katalogu projektu i wywołać polecenie make ARCH=type, gdzie jako type należy podać procesor, na który ma być wygenerowany kod (dla komputera PC ARCH=x86, dla procesora



Rysunek 2. Algorytm działania programu testowego

LPC2142 ARCH=lpc2142 itd.). Aby zaprogramować wybrany mikrokontroler, należy podłączyć odpowiedni programator oraz wydać polecenie make program.

Overclocking mikrokontrolera STM32F107VBT6

Jako pierwszy poddaliśmy testowi mikrokontroler STM32F107VBT6 zainstalowany na płytce STM32Butterfly. Mikrokontroler jest zbudowany w oparciu o architekturę CORTEX-M3 i jego maksymalna częstotliwość taktowania (według danych producenta) wynosi 72 MHz. Układ ma bardzo zaawansowany system dystrybucji sygnału zegarowego (**rysunek 3**), dzięki czemu istnieje możliwość ustawienia mniejszej częstotliwości taktującej jego układy peryferyjne.

STM32 ma również kilka generatorów z pętlą PLL umożliwiających generowanie niezależnych sygnałów zegarowych. Dzięki temu mamy dużą elastyczność konfiguracji, umożliwiającą pracę rdzenia z częstotliwością większą od nominalnej, podczas gdy pozostałe układy peryferyjne możemy skonfigurować tak, aby nie przekraczać ich częstotliwości nominalnych. Próba przetaktowania rdzenia mikrokontrolera odbywać się będzie w sposób programowy, poprzez odpowiednią konfigurację podzielnika PREDIV1 oraz pętli PLL1. W zestawie STM32Butterfly dołączony jest rezonator kwarcowy o częstotliwości 25 MHz, co umożliwia teoretycznie uzyskanie za pomocą pętli PLL częstotliwości 225 MHz. Zmiana częstotliwości taktowania odbywa się poprzez zmianę parametrów przekazanych do funkcji void uc_setup(int divide, int multiply), gdzie divide, jest wartością dzielnika PREDIV1, natomiast multiply wartością mnożnika pętli PLL1. Wyjściowy sygnał pomiarowy OUT został przypisany do linii PE7. Próba będzie polegała na wpisywaniu odpowiednich wartości mnożnika oraz podzielnika, kompilacji, programowaniu mikrokontrolera oraz sprawdzaniu czy przez 20 minut suma MD5 generowana jest prawidłowo, o czym świadczyć będzie obecność przebiegu prostokątnego na wyjściu OUT. Aby wyznaczyć czas wykonania wyliczenia 1 kB bloku danych MD5, należy podzielić okres fali prostokątnej na wyjściu OUT przez dwa. W **tabeli 1** zamieszczono wyniki pomiarów dla STM32.

Najistotniejszymi parametrami są: uzyskana częstotliwość taktowania rdzenia Fclk wyrażona w MHz oraz czas obliczania 1 kB bloku MD5 ExecT wyrażony w milisekundach. Ostatnia kolumna ExecS zawiera czas obliczania 1 kB bloku MD5 w przeliczeniu na 1 MHz taktowania rdzenia, co umożliwi nam późniejsze porównanie wydajności poszczególnych rdzeni. Kolorem białym zaznaczone są wiersze, dla których wartość częstotliwości nie przekracza wartości nominalnej wartości taktowania procesora. Kolorem żółtym zaznaczono wartości częstotliwości przekraczające nominalną wartość taktowania, ale zapewniające stabilną pracę, natomiast kolorem czerwonym zaznaczono wartości częstotliwości, dla których nie udało się uruchomić mikrokontrolera lub nie pracuje on stabilnie. Dla mikrokontrolera STM32F107VBT6 najwyższą częstotliwością pracy, którą udało się uzyskać i przy której rdzeń pracował stabilnie, jest 125 MHz. Przy 150 MHz układ pracuje niestabilnie i zawiesza się po kilkunastu sekundach, natomiast przy częstotliwości 175 MHz , mikrokontrolera w ogólne nie udało się uruchomić. Biorąc pod uwagę, że nominalna częstotliwość zegara wynosi 72 MHz, jest to całkiem dobry wynik.

Możemy zauważyć że przy przełączaniu pomiędzy 25 MHz, a 50 MHz, następuje mniejszy przyrost wydajności, niż wynika to ze zwiększenia częstotliwości taktowania, co wynika z faktu, że dla częstotliwości 50 MHz ustawiamy kontroler pamięci Flash tak, aby wstawiać 3 cykle oczekiwania na dostęp do pamięci. Na **rysunkach 4** i **5** przedstawiono oscylogramy sygnału na wyjściu OUT przy częstotliwości bazowej 25 MHz oraz dla najwyższej (ale niestabilnej pracy) częstotliwości 150 MHz.

MD5 (Message-Digest algorithm 5)

Algorytm z dziedziny kryptografii. Jest to popularna kryptograficzna funkcja skrótu, która z dowolnego ciągu danych generuje 128-bitowy skrót. Algorytm MD5 jest następujący:

- Doklejamy do wiadomości wejściowej bit o wartości 1.
- Doklejamy tyle zer ile trzeba żeby ciąg składał się z 512-bitowych bloków, i ostatniego niepełnego – 448-bitowego.
- Doklejamy 64-bitowy (zaczynając od najmniej znaczącego bitu) licznik oznaczający rozmiar wiadomości. W ten sposób otrzymujemy wiadomość złożoną z 512-bitowych fragmentów.
- Ustawiamy stan początkowy na 0123456789abcdeffedcba9876543210.
- Uruchamiamy na każdym bloku (jest przynajmniej jeden blok nawet dla pustego wejścia) funkcję zmieniającą stan.
- Po przetworzeniu ostatniego bloku zwracamy stan jako obliczony skrót wiadomości.

Funkcja zmiany stanu ma 4 cykle (64 kroki). Stan jest traktowany jako 4 liczby 32-bitowe, i w każdym kroku do którejś z tych liczb dodawany jest jeden z szesnastu 32-bitowych fragmentów bloku wejściowego, pewna stała zależna od numeru kroku oraz pewna prosta funkcja boolowska trzech pozostałych liczb. Następnie liczba ta jest obracana (przesuwana cyklicznie z najstarszymi bitami wsuwanymi w najmłodsze pozycje) o liczbę bitów zależną od kroku, oraz jest dodawana do niej jedna z pozostałych liczb.

Funkcje te to:

W krokach 1 do 16 (cykl 1) funkcja F(x,y,z) = (x and y) or (neg x and z), tzn. jeśli x to y, w przeciwnym wypadku z.

$$F(X,Y,Z) = (X \wedge Y) \vee (\overline{X} \wedge Z)$$

W krokach 17 do 32 (cykl 2) funkcja G(x,y,z) = (x and z) or (y and neg z), tzn. jeśli z to x, w przeciwnym wypadku y.

$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \overline{Z})$$

W krokach 33 do 48 (cykl 3) funkcja H(x,y,z) = (x xor y xor z), tzn. suma argumentów modulo 2 lub innymi słowy - czy występuje nieparzysta liczba jedynek w argumentach.

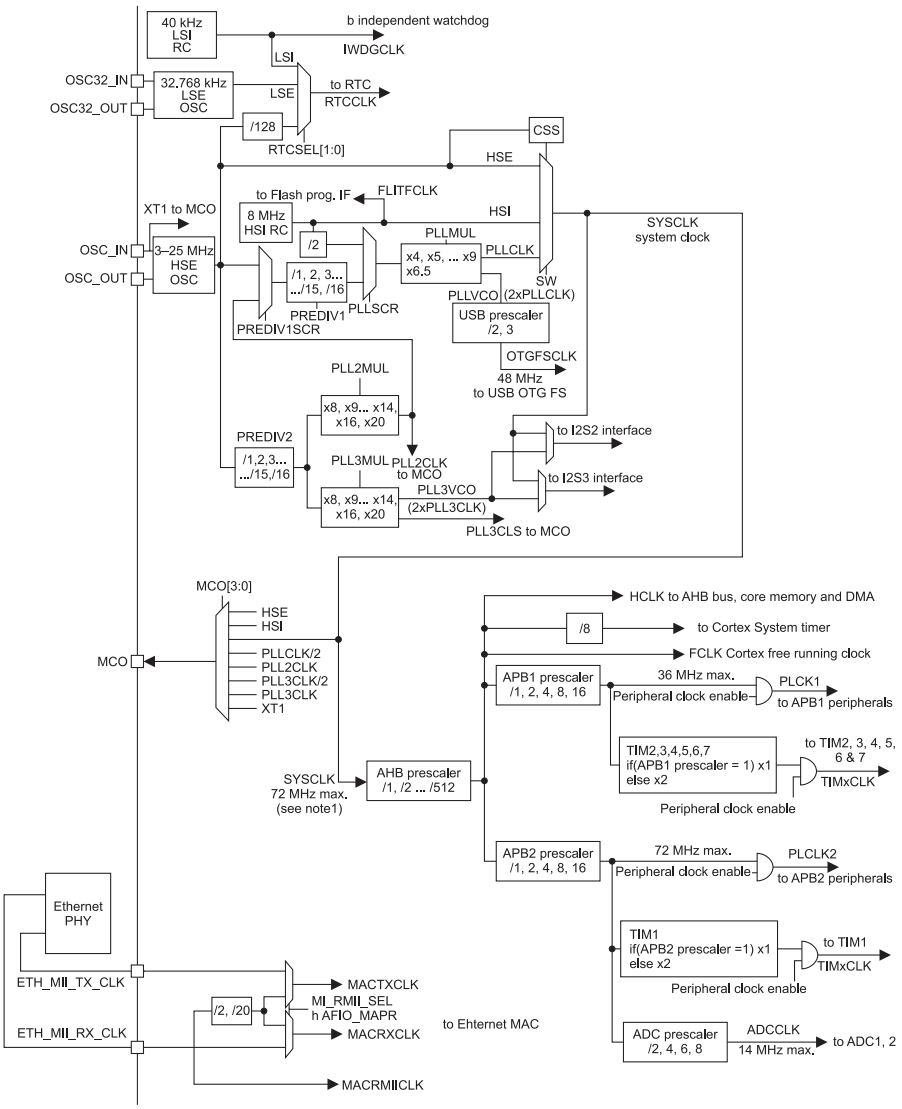
$$H(X,Y,Z) = X \oplus Y \oplus Z$$

W krokach 49 do 64 (cykl 4) funkcja I(x,y,z) = (y xor (x or neg z)), tzn. jeżeli (z = 1 i x = 0) wtedy y, w przeciwnym wypadku nie y.

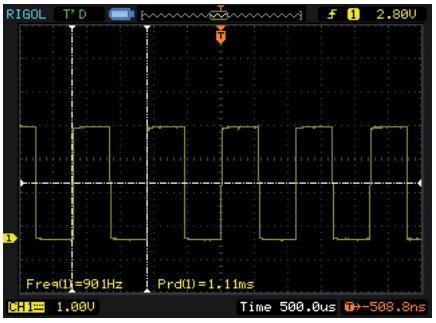
$$I(X,Y,Z) = Y \oplus (X \vee \overline{Z})$$

Tabela 1. Wyniki testów mikrokontrolera STM32

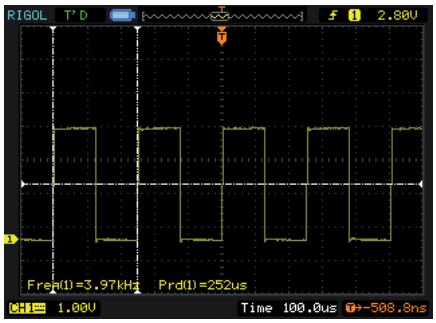
Fosc [MHz]	Div	Mul	Fclk [MHz]	Period [ms]	ExecT [ms]	clk b/a	exec b/a	ExecS [ms/MHz]	Uwagi
8	*	*	8	3,46	1,73	*	*	13,84	Stabilny
25	1	1	25	1,11	0,555	3,125	3,117	13,875	Stabilny
25	2	4	50	0,744	0,372	2,000	1,492	18,6	Stabilny
25	2	6	75	0,496	0,248	1,500	1,500	18,6	Stabilny
25	2	8	100	0,374	0,187	1,333	1,326	18,7	Stabilny
25	2	9	112,5	0,332	0,166	1,125	1,127	18,675	Stabilny
25	1	5	125	0,298	0,149	1,111	1,114	18,625	Stabilny
25	1	6	150	0,252	0,126	1,200	1,183	18,9	Niestabilny po kilku sekundach
25	1	7	175	*	*	*	*		Nie wstaje
								średnia	
								17,477	



Rysunek 3. Układ generowania i dystrybucji zegara w STM32F107



Rysunek 4. Przebieg na wyjściu STM32F107 przy częstotliwości taktowania 25 MHz



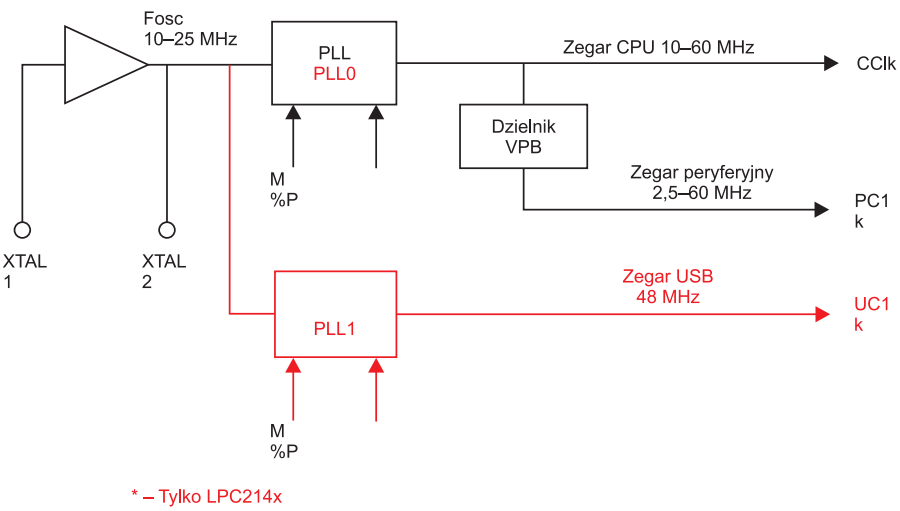
Rysunek 5. Przebieg na wyjściu STM32F107 przy częstotliwości taktowania 150 MHz

Overclocking mikrokontrolera LPC2142

Kolejnym procesorem, który poddaliśmy próbie, jest jeden z pierwszych na rynku przedstawicieli mikrokontrolerów z rdzeniem ARM, dobrze znany mikrokontroler LPC2142. Jako platformę testową wybrano zestaw ZL9ARM+ZL10ARM_2042. Układ ma 16 kB pamięci RAM oraz 64 kB pamięci Flash i nominalnie pracuje z częstotliwością 60 MHz. Jego system dystrybucji sygnału zegarowego jest zdecydowanie prostszy (**rysunek 6**).

System zegarowy posiada oddzielną pętlę PLL dla kontrolera USB, oraz uczęstotliwości taktującej układy peryferyjne w stosunku 1:1, 1:2 oraz 1:4, co umożliwia ustawienie niższej częstotliwości taktowania układów peryferyjnych w stosunku do rdzenia mikrokontrolera. Test będzie polegał na podstawieniu odpowiednich parametrów do funkcji „`void system_periph_init(int mam_cycles, int pll_m);`”, odpowiedzialnej za ustawienie pętli PLL0, kompilacji programu i zaprogramowania układu. Na wyjściu OUT, którym jest linia P0.13 będziemy obserwować przebieg prostokątny świadczący o wyznaczeniu prawidłowej sumy MD5. W **tabeli 2** przedstawiono wyniki pomiarów dla 20-minutowej próby mikrokontrolera LPC2142.

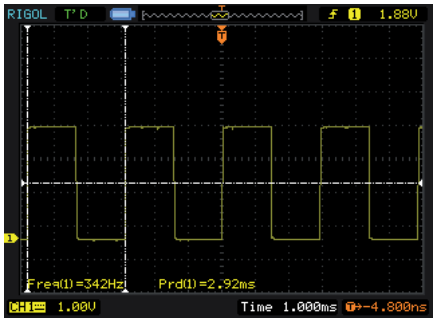
Najwyższa częstotliwość taktowania przy stabilnej pracy rdzenia mikrokontrolera, dla 20-minutowej próby wynosiła 84 MHz. Przy częstotliwości 96 MHz udało się uzyskać pracę jedynie przez okres około kilkunastu sekund po włączeniu. Przy częstotliwości 108 MHz nie udało się w ogóle uruchomić mikrokontrolera. Jak łatwo zauważyć, ten mikrokontroler jest w znacznie mniejszym stopniu podatny na przetaktowanie niż STM32. Możemy również zauważyć, że w przypadku wyznaczania funkcji MD5 pracuje z podobną wydajnością, jak mikrokontroler z nowszym rdzeniem STM32. Na **rysunkach 7** i **8** przedstawiono oscylogramy sygnału na wyjściu OUT dla najniższej oraz najwyższej częstotliwości taktowania, którą udało się uzyskać.



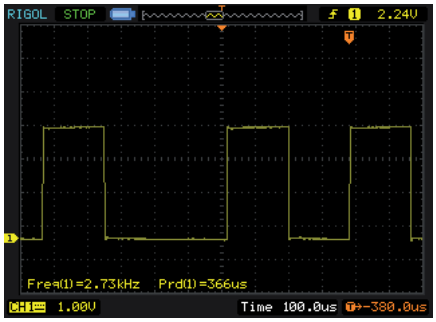
Rysunek 6. Układ generowania i dystrybucji zegara w LPC2142

Tabela 2. Wyniki testów mikrokontrolera LPC2142

Fosc [MHz]	Mul	Fclk [MHz]	Period [ms]	ExecT [ms]	clk b/a	exec b/a	ExecS [ms/MHz]	Note
12	1	12	2,92	1,46	*	*	17,52	Stabilny
12	2	24	1,45	0,725	2,000	2,014	17,4	Stabilny
12	4	48	0,728	0,364	2,000	1,992	17,472	Stabilny
12	5	60	0,584	0,292	1,250	1,247	17,52	Stabilny
12	6	72	0,484	0,242	1,200	1,207	17,424	Stabilny
12	7	84	0,416	0,208	1,167	1,163	17,472	Stabilny
12	8	96	0,366	0,183	1,143	1,137	17,568	Czasami nie działa
12	9	108	*	*	*	*		Nie działa
							średnia	
							17,468	



Rysunek 7. Przebieg na wyjściu LPC2142 przy częstotliwości taktowania 12 MHz



Rysunek 8. Przebieg na wyjściu LPC2142 przy częstotliwości taktowania 96 MHz

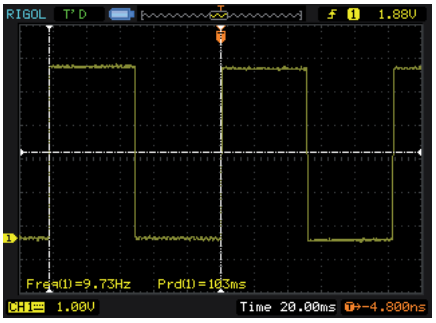
Przy częstotliwości 96 MHz daje się zauważyć pewną niestabilność pracy rdzenia, najprawdopodobniej wynikającą z problemów z pamięcią Flash. Objawia się to nieregularnością przebiegu prostokątnego występującego na wyjściu OUT.

Overclocking mikrokontrolera ATmega32

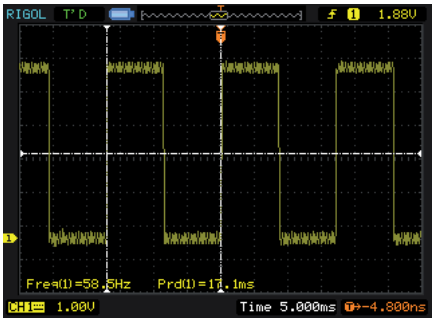
Następnym poddanym próbie układem był przedstawiciel 8-bitowej rodziny AVR – ATmega32, który może być taktowany sygnałem zegarowym o maksymalnej częstotliwości 16 MHz. Jako platformę testową wykorzystano zestaw ZL15AVR. Układ w zasadzie nie ma żadnego zaawansowanego systemu dystrybucji sygnału zegarowego. Zarówno rdzeń, jak i układy peryferyjne mogą być taktowane jedynie z zewnętrznego generatora, wewnętrznego z zewnętrznym rezonatorem kwarcowym lub elementami RC, z jedną z kilku predefiniowanych częstotliwości. Ponieważ nie ma on pętli PLL, zmianę częstotliwości taktowania zrealizowano z wykorzystaniem zewnętrznego generatora dołączanego do linii XTAL1.

Podczas próby nie miałem pod ręką żadnego fabrycznego generatora, dlatego napisałem prosty program (lpcvawe) dla mikrokontrolera LPC2142 (zestaw ZL9+ZL10 ARM), który wykorzystując układ PWM, na linii PWM5 (P0.21) generuje sygnał zegarowy o częstotliwości taktowania podzielonej przez 2. Za pomocą przycisków SW1 oraz SW2 możemy zmieniać częstotliwość z krokiem co 6 MHz. Aktualna wartość częstotliwości generowanego sygnału jest wyświetlana na wyświetlaczu LCD zestawu. Linię P0.21 zestawu ZL9ARM podłączono do nóżki XTAL1 ATmega32. Na wyjściu OUT, które stanowi linia PA2, możemy obserwować przebieg, który informuje o prawidłowej pracy jednostki centralnej. W **tabeli 3** przedstawiono wyniki pomiarów dla 20 minutowej próby.

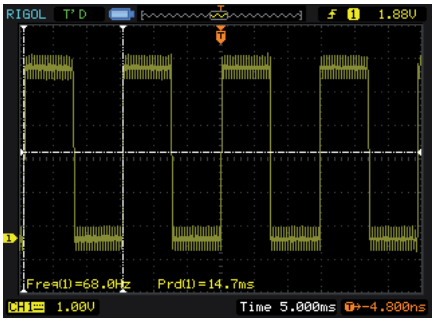
Najwyższą stabilną częstotliwością pracy jaką udało się uzyskać to 36 MHz. Przy 42 MHz układ również pracował poprawnie, jeżeli częstotliwość była zwiększana stopniowo z 36 na 42 MHz. W przypadku, gdy po włączeniu zasilania od razu był podawany sygnał zegarowy o wartości 42 MHz, mikrokontroler w ogóle nie startował. W tym przypadku udało się uzyskać stabilną pracę przy częstotliwości ponad dwukrotnie wyższej od



Rysunek 9. Przebieg na wyjściu ATmega32 przy częstotliwości taktowania 6 MHz



Rysunek 10. Przebieg na wyjściu ATmega32 przy częstotliwości taktowania 36 MHz

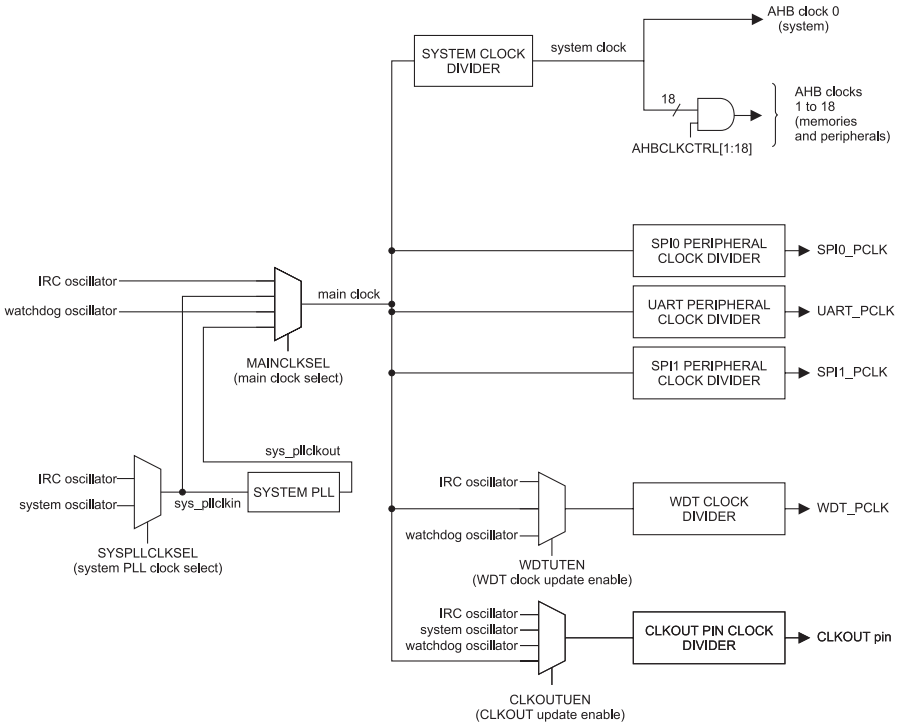


Rysunek 11. Przebieg na wyjściu ATmega32 przy częstotliwości taktowania 48 MHz

nominalnej. To jednak nie dziwi, ponieważ jest to stosunkowo prosty mikrokontroler, charakteryzujący się niewielką wartością częstotliwości nominalnej. Również przy sprawdzeniu czasu wyznaczania funkcji MD5 wi-

Tabela 3. Wyniki testów mikrokontrolera ATmega32

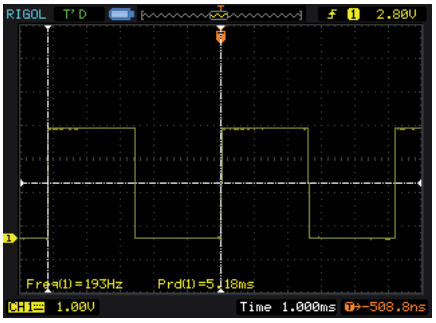
Fclk [MHz]	Period [ms]	ExecT [ms]	clk b/a	exec b/a	ExecS [ms/MHz]	Uwagi
6	103	51,5	*	*	309	Stabilny
12	51,2	25,6	2,000	2,012	307,2	Stabilny
18	34,2	17,1	1,500	1,497	307,8	Stabilny
24	25,6	12,8	1,333	1,336	307,2	Stabilny
36	17,1	8,55	1,500	1,497	307,8	Stabilny
42	14,6	7,3	1,167	1,171	306,6	Nie działa po starcie, jeżeli podnosimy stopniowo częstotliwość to działa
48	*	*	*	*	*	Nie działa
					średnia	
					307,6	



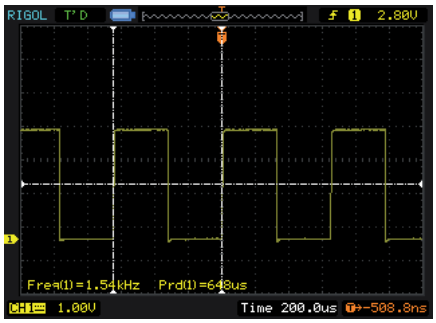
Rysunek 12. Układ generowania i dystrybucji zegara w LPC1114

Tabela 4. Wyniki testów mikrokontrolera LPC1111

Fosc [MHz]	Mul	Fclk [MHz]	Period [ms]	ExecT [ms]	clk b/a	exec b/a	ExecS [ms/MHz]	Uwagi
12	1	12	5,18	2,59	*	*	31,08	Stabilny
12	2	24	2,59	1,295	2,000	2,000	31,08	Stabilny
12	3	36	1,73	0,865	1,500	1,497	31,14	Stabilny
12	4	48	1,29	0,645	1,333	1,341	30,96	Stabilny
12	5	60	1,04	0,52	1,250	1,240	31,2	Stabilny
12	6	72	0,864	0,432	1,200	1,204	31,104	Stabilny
12	7	84	0,74	0,37	1,167	1,168	31,08	Stabilny
12	8	96	0,648	0,324	1,143	1,142	31,104	Zawiesza się po minucie
12	9	108	*	*	*	*	*	Nie działa
							średnia	
							31,094	



Rys. 13. Przebieg na wyjściu LPC1114 przy częstotliwości taktowania 12 MHz



Rys. 14. Przebieg na wyjściu LPC1114 przy częstotliwości taktowania 96 MHz

dzimy, że jednostka centralna charakteryzuje się znacznie mniejszą wydajnością w stosunku do rdzeni ARM. Na **rysunkach 9**, **10**, **11** przedstawiono oscylogramy dla częstotliwości pracy 6 MHz, 36 MHz oraz 42 MHz.

W miarę wzrostu częstotliwości taktowania, możemy zaobserwować coraz wyższy poziom modulacji sygnału przebiegiem zegarowym, nałożonym na przebieg prostokątny. Podobna sytuacja nie występuje na żadnym poprzednio przedstawionym oscylogramie.

Overclocking mikrokontrolera LPC1114

Ostatnim testowanym układem był mikrokontroler z rdzeniem CORTEX-M0, który charakteryzuje się uproszczonym dekoderem instrukcji w stosunku do rdzenia CORTEX-M3. Mikrokontroler może pracować z maksymalnym sygnałem częstotliwości 50 MHz. Układ ma stosunkowo rozbudowane obwody generowania sygnału zegarowego (**rysunek 12**).

Niestety, system rozprowadzania sygnałów zegarowych nie pozwala na zmniejszenie częstotliwości taktowania układów peryferyjnych, zatem będą one pracować przy podwyższonej częstotliwości. Układ ma w zasadzie identyczną pętlę PLL, jak LPC2142. Test będzie polegał na próbie ustawiania częstotliwości taktowania przez zmianę konfiguracji PLL przy wywołaniu funkcji `void pll_setup(int pll_m)`, która jako argument przyjmuje mnożnik podstawowego sygnału zegarowego. Na wyjściu OUT, którym jest linia P0.5, można obserwować występowanie przebiegu prostokątnego, świadczącego o prawidłowym wyznaczeniu sumy MD5. Jako zestaw testowy

wybrano ZL32ARM, w którym jest rezonator kwarcowy o częstotliwości 12 MHz. W **tabeli 4** przedstawiono wyniki dla 20 minutowej próby testów.

Największą wartością stabilnej częstotliwością pracy, którą udało się uzyskać, jest 84 MHz. Zważając na maksymalną, dopuszczalną częstotliwość pracy rdzenia wynoszącą 50 MHz, jest to całkiem dobry wynik. Przy częstotliwości taktowania 84 MHz układ pracował poprawnie przez około 2 minuty, po czym przestał działać. Dla częstotliwości 96 MHz nie udało się w ogóle uruchomić mikrokontrolera. Na **rysunkach 13** i **14** przestawiono oscylogramy na wyjściu OUT przy częstotliwościach taktowania 12 MHz oraz 96 MHz.

Kilka słów na temat wydajności

Do testowania stabilnej pracy mikrokontrolera podczas overclockingu użyto stosunkowo złożonego obliczeniowo algorytm MD5, co umożliwia również porównanie wydajności poszczególnych mikrokontrolerów. Test wydajności odbywa się poprzez porównanie czasu obliczenia sumy MD5 z 1 kB danych. Jako wzorca, w stosunku do którego odbywa się porównanie (współczynnik wydajności 1) wykorzystano komputer PC z procesorem AMD Phenom II, mający pamięci DDR3 taktowane zegarem 1600 MHz. Kod dla komputera PC został skompilowany w trybie 64-bitowym i pracuje pod kontrolą 64-bitowego systemu operacyjnego Linux. Należy tutaj mieć na uwadze, że otrzymane wyniki nie są w pełni miarodajne, ponieważ do porównania wykorzystano tylko jeden algorytm. Niemniej jednak na podstawie tego testu możemy mieć wstępny pogląd na temat stałoprzecinkowej wydajności obliczeniowej poszczególnych rdzeni. Wyniki porównania wydajności przestawiono w **tabeli 5**.

Najbardziej wydajnymi mikrokontrolerami podczas wyznaczania sumy MD5 okazały się STM32F107 z rdzeniem Cortex -M3 oraz LPC2142 ze stosunkowo leciwym rdzeniem ARM7TDMI-S. Według danych producenta rdzeń ARM7TDMI-S powinien mieć wydajność 0,95 DMIPS/MHz, natomiast rdzeń Cortex-M3 powinien mieć wydajność 1,25 DMIPS/MHz. Z tego wynika, że mikrokontroler STM32F107 przy tej samej częstotliwości taktowania powinien być o około 30% szybszy od LPC2142. Jak wynika z tabel, oba mikrokontrolery w zasadzie pracują z podobną wydajnością, co może nieco zaskakiwać. Przyczyn tych rozbieżności możemy się dopatrywać w tym, że test Dryhstone jest testem syntetycznym, wykorzystywanym do celów marketingowych, a rzeczywiste programy z reguły są dużo bardziej złożone. Inną prawdopodobną przyczyną jest mniejsza wydajność kontrolera pamięci Flash w przypadku STM32F107. Mikrokontrolery te charakteryzują się największą otrzymaną wydajnością i są tylko nieco ponad dwa razy mniej wydajne od komputera PC. Rezultat ten jest bardzo dobrym wynikiem wziąwszy pod uwagę, że komputer PC wykorzystuje bardzo szybkie, wydajne pamięci cache, a rdzeń Phenom II jest bardzo skomplikowany gdyż wiele jednostek wykonawczych, dużą pamięć podręczną oraz mechanizm przewidywania skoków.

Na drugim miejscu pod względem wydajności znajduje się mikrokontroler z nowym rdzeniem Cortex-M0, który według producenta charakteryzuje się wydajnością 0,9 DMIPS/MHz. Według tej specyfikacji mikrokontroler ten powinien być niewiele wolniejszy od rdzenia ARM7TDM-S, jednak próba z wykorzystywanym algorytmem pokazuje zupełnie coś innego. Mikrokontroler ten okazał się ponad czterokrotnie mniej wydajny od komputera PC i około dwukrotnie mniej wydajny od mikrokontrolerów Cortex-M3 oraz ARM7TDMI-S. Tak duża różnica wydajności pozwala przypuszczać, że przyczyną tego nie jest kontroler pamięci Flash, a mikrokontroler w przypadku rozbudowanych obliczeń jest dużo mniej wydajny, niż deklaruje to producent. Przyczyną wprowadzenia rdzenia Cortex-M0 była najprawdopodobniej chęć uproszczenia dekodera rozkazów, a zatem minimalizacja zajmowanej powierzchni krzemu i poboru prądu.

Najmniej wydajnym rdzeniem okazał się 8-bitowy mikrokontroler ATmega32 z rdzeniem AVR, co nie jest zaskakujące. Referencyjny algorytm MD5 został opracowany dla architektur 32-bitowych. W samym algorytmie są wykonywane obliczenia na długich liczbach, a poza tym nie ma on rozbudowanej listy rozkazów. ATmega32 dla algorytmu MD5 okazał się ponad 40-krotnie mniej wydajny od komputera PC i około 20-krotnie mniej wydajny od rdzeni Cortex-M3, ARM7TDMI-S. Z testów wydajności dla algorytmu MD5 nasuwa się wniosek, że przy zmianie mikrokontrolera na 32-bitowy, dla skomplikowanych obliczeń może się to okazać bardzo korzystne.

Do porównania wydajności rdzeni producenci mikrokontrolerów bardzo często stosują jednostkę DMIPS. DMIPS jest jednostką określającą liczbę wykonań programu w ciągu 1 sekundy. Wynik obliczeń otrzymujemy w wyniku uruchomienia syntetycznego testu Dryhstone'a. Jest on stosunkowo mało wiarygodny z powodu optymalizacji, które mogą być wygenerowane przez kompilator i generalnie zbyt prosty, aby otrzymać wiarygodne rezultaty. Pomimo, iż został wymyślony ponad 20 lat temu, jest bardzo chętnie stosowany przez firmy z uwagi na duże walory marketingowe.

Tabela 5. Wyniki porównania testowanych mikrokontrolerów

Typ mikroprocesora/mikrokontrolera	ms/MHz	n
Phenom II AMD (AMD64)	7,490	1,000
STM32F107 (Cortex-M3)	17,477	2,333
LPC2142 (ARM7TDMI-S)	17,468	2,332
ATmega32 (AVR)	307,600	41,068
LPC1114 (Cortex-M0)	31,094	4,151

Wnioski

Przeprowadzone testy wykazały, że większość mikrokontrolerów daje się taktować z wyższą częstotliwością od znamionowej. Należy jednak pamiętać, że otrzymane wyniki mogą być specyficzne dla danego egzemplarza mikrokontrolera i praca dla najwyższych, osiągniętych wartości jest ryzykowna w przypadku produktów końcowych. Stosunkowo bezpieczną granicą wydaje się tutaj 1,25 maksymalnej częstotliwości deklarowanej przez producenta. Należy również pamiętać, że należy w tych okolicznościach sprawdzić czy interesujące nas układy peryferyjne będą pracować poprawnie lub (jeżeli istnieje taka możliwość) zmniejszyć częstotliwość ich taktowania poprzez ustawienie podzielnika (nie jest to możliwe dla wszystkich mikrokontrolerów). Najbardziej podatnym na podkręcanie okazał się mikrokontroler ATmega32, który pracował stabilnie z 2,25 maksymalnej częstotliwości zadeklarowanej przez producenta, jednak bardzo mała wydajność obliczeniowa rdzenia AVR, plasuje go na samym końcu naszych zestawień.

Najbardziej podatnym na overclocking i zarazem liderem rankingu pod względem wydajności okazał się mikrokontroler STM32F107VBT6, który pracował stabilnie z 1,7 częstotliwości nominalnej. Stosunkowo rozczarowuje LPC2142. Pracował on stabilnie jedynie dla 1,4 wartości maksymalnej częstotliwości. Mikrokontroler LPC1114 pracował stabilnie z 1,6 wartościczęstotliwości nominalnej, jednak w przypadku skomplikowanych obliczeń rozczarowuje wydajność rdzenia Cortex-M0.

Na koniec uwaga! Należy pamiętać, że moc tracona w procesorze rośnie proporcjonalnie do częstotliwości taktowania napięcia zasilania. Na ten aspekt przetaktowywania należy też zwrócić uwagę, aby nie przegrzać procesora.

Lucjan Bryndza, EP
lucjan.bryndza@ep.com.pl

Literatura:
http://www.goodram.com/news,91,f3ea7.html
http://www. pugetsystems.com
http://pl.wikipedia.org/wiki/MD5