VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Seminář VHDL – Projekt

Světelné noviny

8. května 2022 Dominik Horký (xhorky32)

1 Zadání projektu

Implementujte světelné noviny na maticovém displeji pouze s využitím FPGA. Aplikace se bude ovládat pomocí vestavěné vnitřního stavového automatu. Zobrazování bude realizováno pomocí maticového displeje 16x8 bodů, který bude možné zapůjčit. Na displeji se zobrazí obrázek, který se bude posouvat (rotovat) doleva či doprava. Směr se bude po třech iteracích měnit a na závěr proběhne jednoduchá animace.

1.1 Implementace a dodržení zadání

Při implementaci jsem narazil na několik problémů, díky kterým jsem zcela nedodržel zadání. Také jsem musel změnit implementaci úkolů, jelikož nefungovaly dle mých představ.

1.1.1 Úkol 1: Funkce GETCOLUMN

Při práci na projektu jsem narazil na problém při generování sloupců a získávání dat z paměti ROM. Funkce GETCOLUMN, implementovaná v prvním úkolu vracela špatná data, resp. špatný kus dat a bylo nutné ji mírně poupravit.

1.1.2 Úkol 2: Čítač

V čítači jsem používal pro hodnoty výčtových typů typ `time`, díky kterému však nešel poté projekt přeložit (což ale v druhém úkolu nebylo potřeba). Místo něj jsem použil jednoduchý `integer`. Dále jsem s těmito výčtovými typy (vstupní, výstupní perioda) špatně pracoval, což bylo nutné taky opravit.

1.1.3 Nedodržení zadání - posuvy

Jelikož jsem podcenil obtížnost a časovou náročnost tohoto projektu, nepodařilo se mi implementovat posuvy obrázku a animace jak je specifikováno dle zadání. Nechci však odevzdávat zcela nefunkční projekt, a proto jsem vymyslel náhradní řešení, kterým je implementace posuvů pomocí nahraného obrázku a jeho posunutých variant do paměti ROM a následné čtení z paměti. V řadiči se tak poté pouze inkrementuje adresa, a posuvy obrázku a animace je řešeno pouhým čtením z paměti ROM.

Tyto posuvy jsou také realizované pouze 2x, nikoliv 3x, jak je vyžadováno v zadání a to z důvodu omezeného řešení těchto posuvů (čtení dat z ROM) a omezené paměti FITKitu.

Díky takovému způsobu implementace však nepotřebuji pracovat s navrženým automatem ani se stavy sousedů, které jsou uložené v entitě sloupce.

1.1.4 Implementace - co je tedy splněno?

Hlavní částí projektu je řadič, který pracuje s entitami časovače, (automatu), paměti ROM a sloupce. Poté je v řadiči ještě proces, který pracuje s pomocnými signály a čítačem. Úplně na konci se pak přiřazují hodnoty do signálů `A` a `R`, které určují obsah displeje, resp. které LED (signál R) ve vybraném sloupci (signál A) budou svítit a které nikoliv.

Samotná entita řadiče je vytvořena v souboru `fpga_inst.vhd`, která mapuje jeho výstupy do signálů určujících aktuální obsah displeje.

Koncept 16 samostatných sloupců tvořených konstrukcí for-generate byl dodržen, stejně tak jako entita sloupce (i když se v něm některé signály nepoužívají), časovače či paměť ROM. Vytvořen je i Moorův konečný automat (entita fsm), který však není potřeba vzhledem ke mému omezenému řešení projektu.

2 Odůvodnění použitých konstant

V projektu jsem použil několik konstant. Některé z nich jsou posány i ve zdrojovém kódu, všechny však uvedu a popíšu i zde.

2.1 Konstanty časovače

Entita časovače pracuje s dvěmi výčtovými typy - `CLK_PERIOD` a `OUT_PERIOD`. Oba tyto typy jsou datového typu integer a reprezentují čas, resp. časovou jednotku v nanosekundách.

2.1.1 Vstupní perioda CLK_PERIOD

Oba časovače (v projektu - řadiči mám dva) mají tuto hodnotu nastavenou na 40. Je to z toho důvodu, že se jedná o frekvenci hodinového signálu, kterou generuje FPGA na FITKitu a s kterou tedy musím pracovat. Tato hodnota je uvedená v souboru `fpga.vhd`, konkrétně na řádku 104, v proměnné `dcm_freq`, hodnota činí 25 MHz.

Jelikož `CLK_PERIOD` reprezentuje periodu - čas, pomocí jednoduchého vztahu:

$$t = \frac{1}{F}\ [s]$$

$$CLK\_PERIOD = \frac{1}{25000000}$$

$$CLK\_PERIOD = 0,00000004 * (10^9)\ [s]$$

$$CLK\_PERIOD = 40\ [ns]$$

2.1.2 Výstupní perioda OUT_PERIOD

Zde už se hodnoty v časovačích liší, jelikož každý používám na něco jiného.
V časovači `led_change`, určujícím jak rychle se mění (zapínají a vypínají) LED v displeji se používá hodnota 17 362, protože se zde zohledňuje frekvence hodinového signálu FPGA (dcm_freq, v Hz), počet sloupců dipsleje a zvolená obnovovací frekvence displeje (v Hz). Počet sloupců je daný (hadwarem, dodaným displejem), frekvenci 90 Hz jsem si zvolil, protože jde o jednu z dnes běžně používaných frekvencí.

$$OUT\_PERIOD = \frac{\frac{25000000}{16}}{90}$$

$$OUT\_PERIOD = 17362\ [ns]$$

Výsledná hodnota nevyšla v celém čísle, je zaokrouhlena nahoru.

V časovačí `column_change`, který určuje, jak rychle se budou sloupce v displeji měnit (určuje rychlost animace) se používá hodnota 5 078 125. Tato hodnota je tvořena z časování změn sloupce (nebo také rychlosti, v ns), vstupní periody (ns) a počtu sloupců v displeji.

$$OUT\_PERIOD = \frac{\frac{3250000000}{16}}{40}$$

$$OUT\_PERIOD = \frac{203125000}{40}$$

$$OUT\_PERIOD = 5078125\ [ns]$$

První hodnota, časování změn sloupců je nastavena na 203 125 000 nanosekund, respektive 0,203125 sekund. To znamená, že všechny sloupce se budou měnit (”aktualizovat”) každou tuto hodnotu. Za jednu sekundu se tedy provede zhruba 5 změn - pohybů na displeji.

2.1.3 Ostatní

Pro velikost dat (signály `current_data` a `ROM_data` v řadiči, popř. `DATA` v ROM) se používaly vektory o velikosti 128 bitů, protože právě tolik bitů je potřeba na pokrytí všech LED žárovek v 16x8 displeji.

Čítač v řadiči je velký 4 bity, protože doslova "kopíruje" signál A (signál, který určuje momentálně vybraný sloupec v displeji) a jeho stav, tudíž musí mít stejnou velikost.

Signál `current_col` v řadiči řeší současně vybraný sloupec a jelikož jde o integer, jeho velikost musí být v rozsahu počtu sloupců displeje

Signál `current_address` v řadiči určuje, jaká data z paměti ROM jsou současně vybrána (index pole). Jelikož se v implementaci liším od zadání a paměť mám více naplněnou, tento integer má jednoduše rozsah do 1000 (aby nebylo nutné při práci s pamětí neustále tyto hodnoty měnit, nepředpokládám, že moje paměť ROM bude pracovat s více jak 1000 různými daty-obrázky).

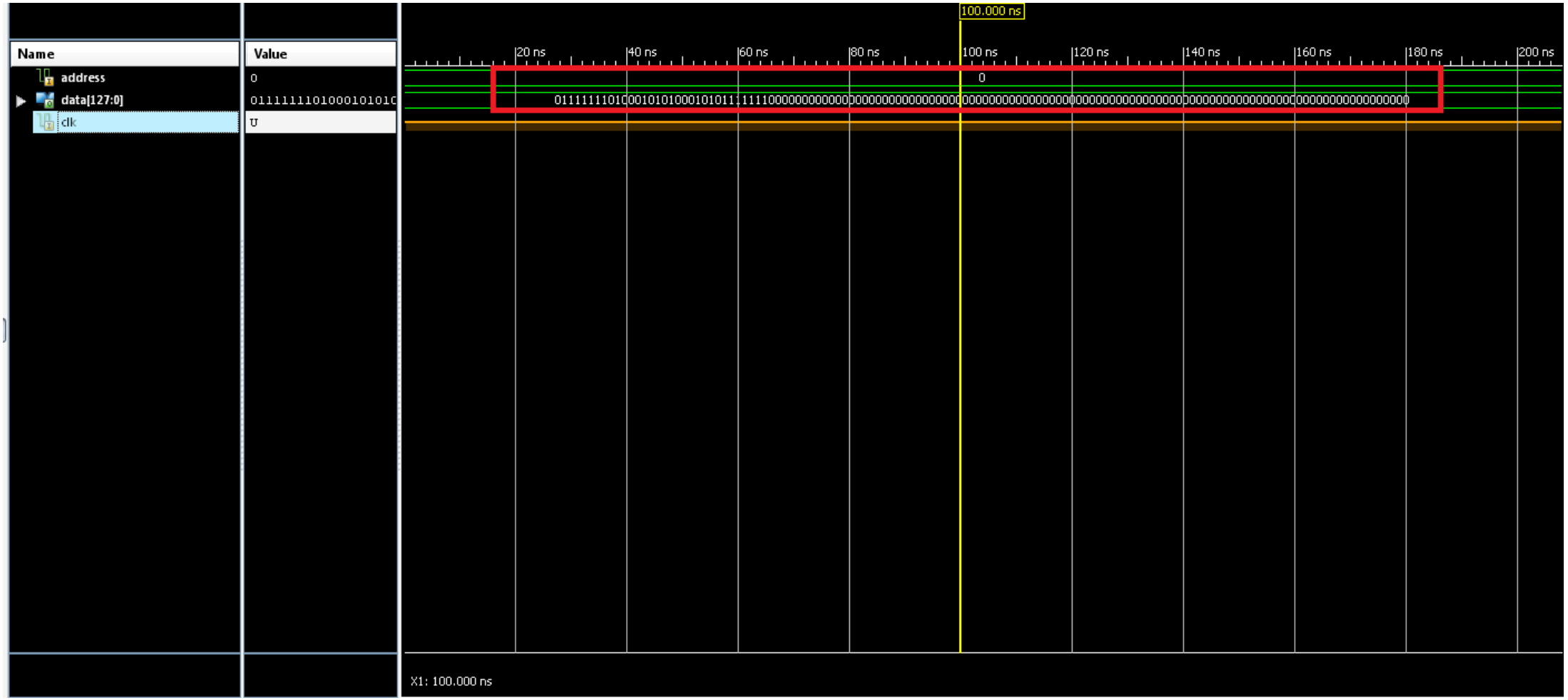
Dále (v řadiči) při práci s funkcí GETCOLUMN všude uvádím hodnotu 8, je to jednoduše proto, že funkce požaduje počet řádků a dodaný displej jich má 8.

3 Odkaz na video implementace

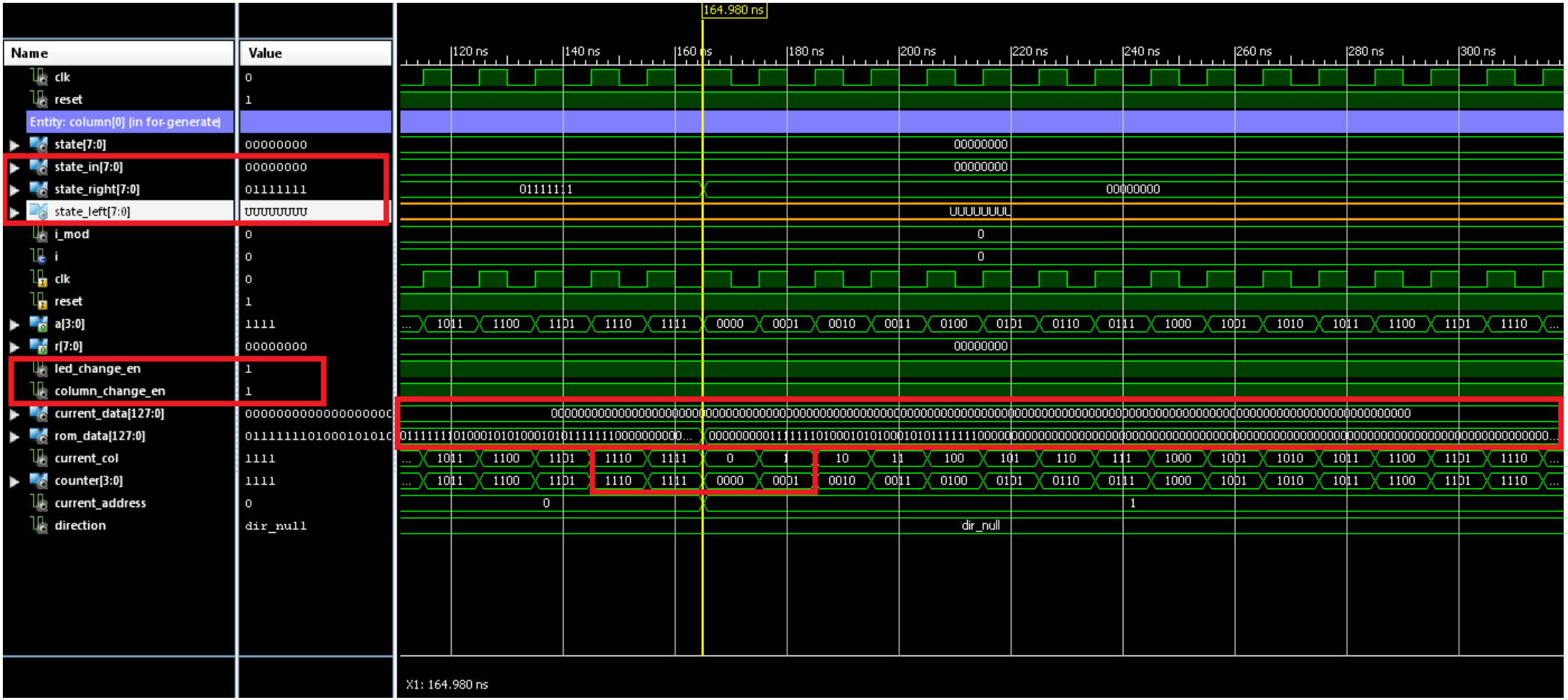
Video je nahrané na Google Drive a ukazuje chování displeje po nahrání projektu do FITKitu přes fkflash. Jako obrázek, který se pohybuje jsem použil panáka chodícího tam a zpět. Na závěr se spustí animace střídající se šachovnice. Odkaz: https://drive.google.com/file/d/15DbQy4IKyUPB5_NzilrOc9D5KsaG2pb3/view?usp=sharing.

4 Výstupy požadovaných simulací

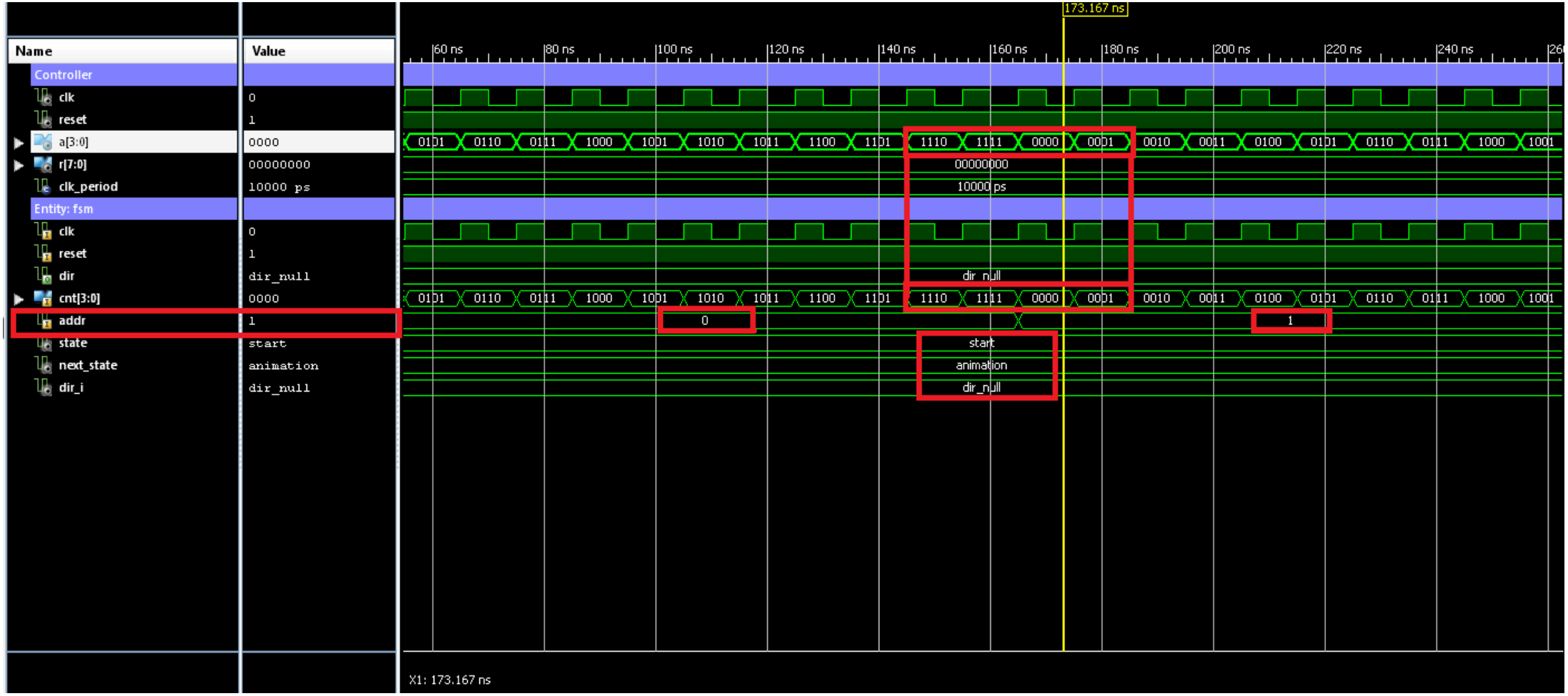
Na dalších stránkách budou snímky obrazovky ze simulací požadovaných dle zadání.



Obrázek 1: Simulace paměti ROM



Obrázek 2: Simulace entity sloupce



Obrázek 3: Simulace řadiče FSM

Obrázek 4: Simulace celého projektu

4.1 Simulace paměti ROM

Simulace paměti ukazuje, že funguje správně, jelikož je nastavená adresa na nulu, tedy první prvek pole-paměti a v signálu data jsou správná odpovídající data pro daný index.

4.2 Simulace entity sloupce

Zde je vidět, že se nastavuje hodnota pravého i levého souseda. Zároveň jsem zde vystihl zrovna moment, kdy se mění obsah displeje i dalšího sloupce zároveň (tj. všechny sloupce byly zobrazeny, zobrazuje se další obrázek). To poznáme diky aktivnímu signálu `led_change_en` a `column_change_en`. Také je zde vidět, že počítač sloupců i počítač současného sloupce fungují správně.

4.3 Simulace řadiče FSM

Jelikož automat v mém řešení není potřebný, není divu, že stavy se zde nemění. Používám pouze dva stavy, které jsou spíše kosmetické, tudíž se zde ani nemění. Zároveň je zde vidět, že směr je nastavený na `DIR_NULL`, což znamená, že se obrázek nehýbe. Jak jsem dříve zmiňoval, projekt funguje tak, že pohyby jsou realizovány jako čtení z paměti RAM a inkrementace adresy, tudíž směr vůbec určovat nepotřebuji. Je zde i vidět inkrementace adresy po rozsvícení požadovaných LED ve všech sloupích displeje. Dále je zde vidět správná funkčnost čítače, který je synchronizovaný se signálem A.

4.4 Simulace celého projektu

Zde vidíme hodnoty vstupní a výstupní periody obou časovačů. V entitě automatu je směr nastavený na `DIR_NULL`, což je realizováno kvůli způsobu implementace. Je zde vidět i informace o současném a dalším stavu v automatu. V bodě, kam míři šipka se právě mění obrázek, resp. data z paměti ROM, jelikož byla inkrementována-změněna adresa. Zároveň jako další důkaz, že se mění obrázek, je to, že entity obou časovačů, resp. jejich `EN` signály jsou aktivní.