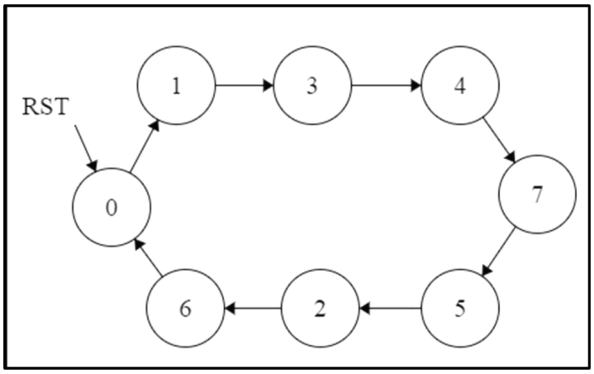
Véges állapotú gépek (FSM) tervezése 3.
(a tavalyi első házi feladat)

F1. A feladat olyan 8 állapotú állapotgépek tervezését írja elő, amelyek a kimenetükön előírt sorrendben jelenítik meg az oktális számrendszer számértékeit. Minden hallgató rendelkezik egy egyéni, un. DIGIT kóddal, amely egy 7 számjegyű számsorozat. Ezt az első pozícióban a 0 értékkel kiegészítve áll elő a tervezendő feladat állapotváltozásainak sorrendjét leíró számsorozat. Ezt az állapotgépet 3 különböző módon kell megtervezni és a tervezés eredményét szimulációval ellenőrizni. Minden tervezendő verzió legyen egy-egy önálló modulban realizálva. Az interfészek neveit a részfeladatok specifikációja tartalmazza. Legyen a leírás további részében a minta DIGIT kód 1347526 (7 különböző számjegy, az egyes állapotok azonosítói, kódjai egy adott sorrendben). Ezt kiegészítjük a kezdeti 0 számjeggyel (01347526) így megkapjuk a működést jellemző állapotsorozatot, ami egy RST reset jel után az alábbi időbeli állapotátmeneti sorrendet eredményezi a kikapcsolásig vagy újabb RST reset pulzusig.



01347526013475260134752601347526013475260134752601347526013475260134752601347…

F1.a *Hagyományos (kézi) tervezési mód.* A véges állapotú vezérlő megtervezése hagyományos, papír-ceruza módszerrel. A tervezendő modul neve legyen MAN_FSM.v, a modul kimenete jele legyen a 3 bites MAN_OUT[2:0]. Első lépésként az állapotátmenetek felírása után megkapjuk a tervezendő egység állapotátmeneti függvényét definiáló táblázatot. Az állapotátmeneti függvény egy 3 bemenetű, 3 kimenetű logikai függvény, az aktuális állapotból előállítja a következő állapot kódját. A példa sorozatra oktális számrendszerben ábrázolva a következőt kapjuk:

Jelenlegi állapot	Következő állapot
0	1
1	3
3	4
4	7
7	5
5	2
2	6
6	0

Az állapotregiszter egy hárombites regiszter, a modulon belüli azonosítója legyen state[2:0]. A következő állapotot azonosító változó neve legyen next_state[2:0].

Írja fel a kiinduló logikai egyenleteket, amelyek segítségével a state[2:0] 3 bites állapotváltozó jelenlegi értéke alapján a következő állapot értékét meghatározó 3 bit, azaz a next_state[2], a next_state[1] és a next_state[0] egyenként előállítható. Tehát három darab, az állapot átmeneti előírások felhasználásával előállított egyedi 3 változós logikai függvényt várunk, amelyeket tetszőleges módszerrel egyszerűbb alakra hozva minimalizálunk. A végeredményként kapott minimalizált függvényeket a Verilog HDL nyelv logikai operátorainak (&, |, ^, ~) használatával írjuk be a forráskódba. Ha vannak redundáns (többször is előforduló) kifejezések a 3 önálló kimeneti függvény előállításában, akkor azok kihasználhatóságát is vegye figyelembe (minimalizálás többszörös kimenete esetén)!

F1.b ***Általános tervezési mód.*** A 8 állapotú állapotgép tervezése a sorrendi hálózatok tervezésére megismert általános módszer szerint. A modul neve legyen STD_FSM.v, a 3 bites kimeneti jel neve legyen STD_OUT[2:0]. A modulon belül az állapotregiszter neve legyen state[2:0], a következő állapotot reprezentáló jel neve next_state[2:0]. A tárgy előadásain, gyakorlatain és laboratóriumi foglalkozásain megismert, véges állapotú állapotgépek tervezésére javasolt tervezői mintát követve Verilog HDL nyelven specifikálja a személyes DIGIT kódban megadott mintasorozat szerint működő állapotgépet. Az állapotok szimbolikus neve legyen START, A, B, C, D, E, F és G. Rendelje hozzá a szimbolikus nevekhez a DIGIT kód által előírt konkrét numerikus kódok értékét és specifikálja az állapotgép működését!

F1.c ***Indirekt tervezési mód.*** A 8 állapotú állapotgép tervezése indirekt módon. A modul neve IND_FSM.v, a 3 bites kimeneti jel neve legyen IND_OUT[2:0]. A terv alapja egy 3 bites bináris számláló (lehetne bármilyen más számláló is, pl. Gray). A számláló a 0 kezdeti értékből indulva, sorban végig lépked a 8 lehetséges állapoton, bináris esetben a 012345670123… sorrendben. A modul tartalmaz egy kimeneti átkódoló hálózatot, amely a fenti (normál numerikus) sorrendet leképezi a DIGIT kódban megadott számsorrendre. Tehát, míg a számláló a saját 012345670123… sorrendjét generálja, a kimeneten ennek alapján (a mintának megfelelően) a 013475260134… sorrend jelenik meg. A cnt[2:0] kódokat a kimeneti IND_OUT[2:0] kódokra leképező átkódolót tetszőleges, a tárgy keretében tanult módszerrel megtervezheti. Törekedjen a szép, esetleg öndokumentáló jellegű Verilog HDL kódolási stílus választására, használatára. De természetesen az elsődleges feladat a funkcionálisan helyes átkódoló megtervezése!