```
    TITLE "Trigger Emulator Sequence Controller";

% INCLUDE "thcm.inc"; %

% ******************************************************************************
*
*                        Trigger Emulator Sequencer, w.r.k - 1998
*
******************************************************************************
* CTRLDAT[] selects the operations of the sequencer.
* It is a feedback of the upper 16 bits of the sequencer memory.
* 4 bits are used for control function, 12 bits for Loop count.
* Master clock input = 53 MHz
*
*   CTRLDAT[]       Operation           Load Count  Decr. Count  Load Addr. Inc. Addr
*
*       NOOP        No special op           NO      NO      NO      YES
*       RUN     Execute             NO      NO      NO      YES
*       LoopStart   Start of Loop           YES(1)      NO      NO      YES
*       LoopEnd     End of Loop             NO      YES     YES(2)      YES(2)
*       FIFOEND     End of sequence         NO      NO      YES(3)      NO
*
* Note 1: Only load Loop Counter if Loop Counter = 0
* Note 2: Only reload Address Counter if Loop Counter > 0, else inc. address
* Note 3: Only load/reset Address Counter to 0 if EmuRun == 1

    7/18/98 - NGW

    1. Changed the ClkDiv counter to a gray code counter due to glitches on the outputs of the
    Phase[] flops. Timing remains the same, just the Phase[] value changes.

    2. Changed the ClkDiv[].clrn timing. Problem was that the resultant "zero" from the clear
        was causing a setup violation in relation to the ClkDiv[] clock.
%

% ********** these are the control words from the emulator memory ********** %
CONSTANT NOOP = 0;
CONSTANT LoopStart = 1;
CONSTANT LoopEnd = 2;
CONSTANT FifoEnd = 4;
CONSTANT TrigOut = 8;


SUBDESIGN   SeqCtrl (
    CTRLDAT[3..0]       :INPUT;  % sequencer memory bits [94..92] %
    CTRLVAL[11..0]      :INPUT;  % sequencer memory bits [91..80] contain loop count %
    MemAddr[12..0]      :OUTPUT; % Sequencer memory address lines %
    MuxWrN[1..0]        :OUTPUT; % Latch sequencer output into multiplexer %
```

```
    RFCLK               :INPUT;  % 53 MHz RF clock %
    BSYNC               :INPUT;  % Beam Sync, i.e. 132 nsec tick %
    EmuEnabled          :INPUT;  % Emulation mode enabled %
    Contin              :INPUT;  % Run Emulation in continuous mode %
    EmuTrig             :INPUT;  % Trigger start of emulation cycle %
    EmuRun              :OUTPUT; % Emulator in RUN mode %
    SeqMemOe            :OUTPUT; % Sequencer memory Output Enable %
    out_trig            :OUTPUT; % output trigger to front panel %
    /reset              : INPUT ; % reset signal for entire board %
)

VARIABLE

    SeqSt           : MACHINE WITH STATES
                         ( SeqIdle,
                           SeqRun,
                           SeqLoop,
                           SeqDone);


    LpCounter[11..0]    : DFF; % loop counter %
    LoopStat            : DFF;
    AddrCt[12..0]       : DFF; % Address Counter - output goes to MemAddr[11..0] %
    AdLatch[12..0]      : DFF; % Latch start of loop address here %
    ClkDiv[3..0]        : DFF; % RFCLOCK count within BSYNC tick %
    Phase[3..0]         : DFF; % synchronize the clock divider out %
    CLKRES              : DFF; % syncronize phase to BSYNC %
    EmuFF               : DFF; % Emulation active (RUN) ff %


    LoopCtLd            : NODE;
    AddrInc             : NODE;
    AddrPop             : NODE;
    AddrRes             : NODE;
    EmuDone             : NODE;
    templatch           :DFF;

BEGIN

DEFAULTS
    LpCounter[11..0].clrn   = VCC;
    EmuDone     = GND;
END DEFAULTS;


templatch.clk = BSYNC;
templatch.d = GND;
templatch.prn = !(BSYNC AND (templatch.q == GND)) AND /reset ;
```

```
EmuFF.clk = EmuTrig;
EmuFF.D = VCC;
EmuFF.clrn = EmuEnabled AND (!(EmuDone AND !Contin)) AND /reset;

EmuRun = EmuFF.q;

--ClkDiv[].clrn = !CLKRES.q;
ClkDiv[].clrn = !(BSYNC AND  (templatch.q == 1));

LoopStat.clk = RFCLK;
LoopStat.d   = !(LpCounter[] == 0);
LoopStat.clrn = /reset;

--CLKRES.d   = VCC;
--CLKRES.clk = BSYNC;
--CLKRES.clrn = !(BSYNC & CLKRES.q);

CLKRES.d   = GND;
CLKRES.clk = BSYNC;
CLKRES.prn = CLKRES.q;
CLKRES.clrn = /reset;


Phase[].clk = RFCLK;
PHASE[].d   = ClkDiv[].q;
Phase[].clrn = /reset;

AddrCt[].clrn = (!AddrRes) AND /reset;
--AddrCt[].clk = ((AddrInc # AddrPop) & (Phase[] == 5));
% had to change from 5 to 7 once the gray code counter was implemented %

AddrCt[].clk = ((AddrInc # AddrPop) & (Phase[] == 7));


% AdLatch[].clk = ((ClkDiv[] == 5) & (CTRLDAT[3..0] == LoopStart)); %
CASE ClkDiv[].q IS
    WHEN B"0000" =>
        ClkDiv[].d = B"0001";
    WHEN B"0001" =>
        ClkDiv[].d = B"0011";
    WHEN B"0011" =>
        ClkDiv[].d = B"0010";
    WHEN B"0010" =>
        ClkDiv[].d = B"0110";
    WHEN B"0110" =>
        ClkDiv[].d = B"0111";
    WHEN B"0111" =>
```

```
        ClkDiv[].d = B"0101";
    WHEN B"0101" =>
        ClkDiv[].d = B"0100";
    WHEN B"0100" =>
        ClkDiv[].d = B"1100";
    WHEN B"1100" =>
        ClkDiv[].d = B"1101";
    WHEN B"1101" =>
        ClkDiv[].d = B"1111";
    WHEN B"1111" =>
        ClkDiv[].d = B"1110";
    WHEN B"1110" =>
        ClkDiv[].d = B"1010";
    WHEN B"1010" =>
        ClkDiv[].d = B"1011";
    WHEN B"1011" =>
        ClkDiv[].d = B"1001";
    WHEN B"1001" =>
        ClkDiv[].d = B"1000";
    WHEN B"1000" =>
        ClkDiv[].d = B"0000";
END CASE;

AdLatch[].clk = BSYNC;
ClkDiv[].clk = RFCLK;

% removed due to gray code counter implementation %
--ClkDiv[].d = ClkDiv[].q + 1;
% Changed from 3 to 2 when gray code counter implemented %
--LpCounter[].clk   = ((Phase[] == 3) & (CTRLDAT[3..0] == LoopStart));

LpCounter[].clk     = ((Phase[] == 2) & (CTRLDAT[3..0] == LoopStart));
LpCounter[].clrn    = EmuRun AND /reset;


SeqSt.clk           = BSYNC;
SeqSt.reset         = !EmuRun;

%*******************************************************************************************
*
*                   Emulator Sequencer state machine: SeqSt
*
*******************************************************************************************%

CASE ( SeqSt ) IS

    WHEN SeqIdle =>
        IF (EmuRun) THEN
```

```
            SeqSt       = SeqRun;
        ELSE
            SeqSt       = SeqIdle;
        END IF;

    WHEN SeqRun =>

        CASE (CTRLDAT[3..0]) IS
            WHEN NOOP =>
                SeqSt       = SeqRun;

            WHEN FifoEnd =>
                SeqSt = SeqDone;

            WHEN LoopStart =>
                SeqSt = SeqLoop;

        END CASE;

    WHEN SeqLoop =>

        CASE (CTRLDAT[3..0]) IS

            WHEN NOOP =>
                SeqSt = SeqLoop;    % Hold State %

            WHEN LoopEnd =>
                SeqSt = SeqRun;

            WHEN FifoEnd =>
                SeqSt = SeqDone;

        END CASE;

    WHEN SeqDone =>
        SeqSt = SeqIdle;                % In continuous mode go back to zero %

END CASE; % end of state machine %

EmuDone = (SeqSt == SeqDone);

AddrRes = (EmuDone # !EmuRun);

AddrInc = !((EmuRun == GND) #
           ((SeqSt == SeqLoop) & (CTRLDAT[3..0] == LoopEnd) & (LoopStat == VCC)));

AddrPop = ((SeqSt == SeqLoop) & (CTRLDAT[3..0] == LoopEnd) & (LoopStat == VCC));
```

```
% LoopCtDec = ((SeqSt == SeqRun) & (CTRLDAT[3..0] == LoopStart) & (LoopStat == VCC)); %

LoopCtLd  = ((SeqSt == SeqRun) & (CTRLDAT[3..0] == LoopStart) & (LoopStat == GND));

IF (AddrInc == VCC) THEN
    AddrCt[12..0].d = AddrCt[12..0].q +1;   % increment address %
  ELSE IF (AddrPop == VCC) THEN
    AddrCt[12..0].d = AdLatch[12..0].q;     % pop stack address %
  ELSE
    AddrCt[12..0].d = AddrCt[12..0].q;      % hold address %
  END IF;
END IF;

IF (LoopCtLd == VCC) THEN
    LpCounter[11..0].d = CTRLVAL[11..0];
ELSE
    LpCounter[11..0].d = LpCounter[11..0].q - 1;
END IF;

IF(CTRLDAT[3..0] == LoopStart) THEN
    AdLatch[12..0].d = AddrCt[12..0].q;
ELSE
    AdLatch[12..0].d = AdLatch[12..0].q;
END IF;

% sequencer addresses are driven by the address counter %
MemAddr[12..0] = AddrCt[12..0].q;

% latch output into mux on second rfclock tick %
MuxWrN[0] = (AddrInc # AddrPop) & (Phase[] == 1);
% changed from 4 to 6 due to gray code counter %
--MuxWrN[1] = (AddrInc # AddrPop) & (Phase[] == 4);
MuxWrN[1] = (AddrInc # AddrPop) & (Phase[] == 6);

% Sequencer output enable allways when state != Idle %
SeqMemOe = (SeqSt != SeqIdle);

% output trigger is generated by emulation memory control lines and the emulation
    mode enabled %
out_trig = ((CTRLDAT[] == TrigOut) AND EmuEnabled);

END;
```