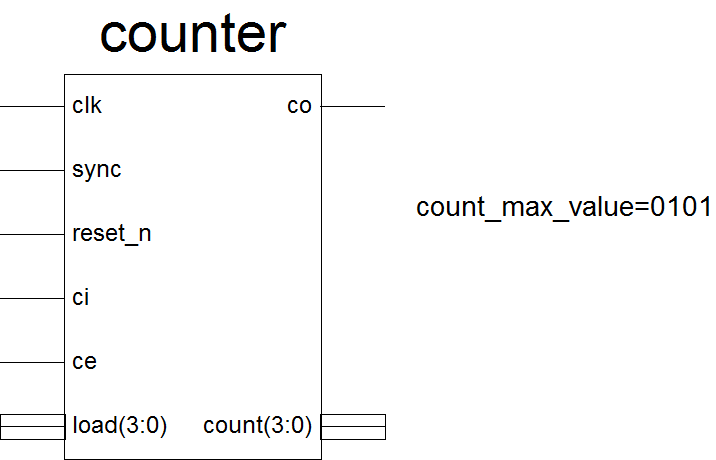
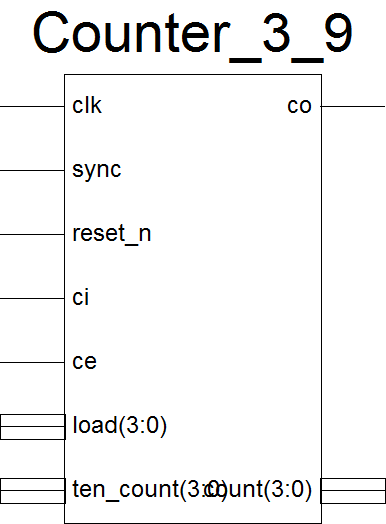
Counter

Es wurden zwei Zählermodule umgesetzt. Dies hat den Grund, dass 5 der 6 Zähler einen fixen Endwert besitzen und einer einen Variablen.

# Inputs:

|  |  |
| --- | --- |
| clk : in STD\_LOGIC; | Takt (Duty-Cycle 50%) |
| ce : in STD\_LOGIC; | Zählerfreigabe ( input für Sekundentakt) |
| reset\_n: in STD\_LOGIC; | Synchroner Reset |
| sync : out STD\_LOGIC; | Zeit Synchronisierungsignal: ‘1‘ während einem Takt. |
| ci : in STD\_LOGIC; | Übertragseingang |
| load : out STD\_LOGIC\_VECTOR (3 downto 0); | Ladewert wenn sync=‘1‘ und falling\_edge(clk) |
| ten\_count : in STD\_LOGIC\_VECTOR (3 downto 0); | Zählerstand Zehnerstelle für Umschaltung 3 und 9 Stunden Zählung |

# Outputs:

|  |  |
| --- | --- |
| co : out STD\_LOGIC; | Übertragsausgang |
| count : out STD\_LOGIC\_VECTOR (3 downto 0); | Zählerausgang |

# Generic

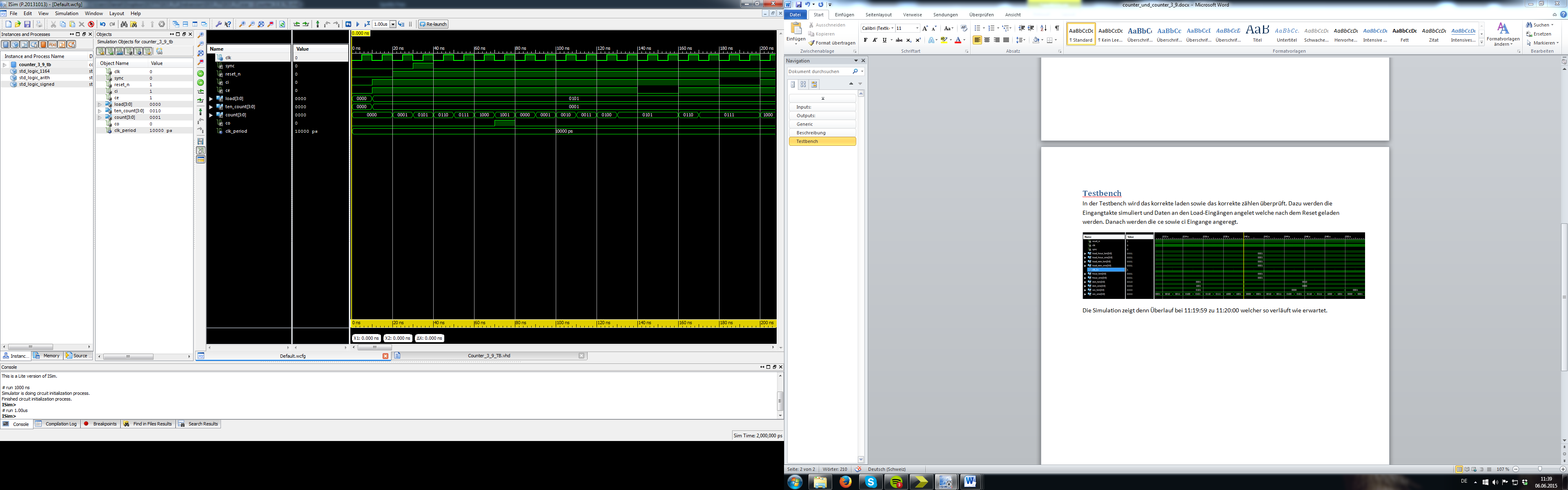
|  |  |
| --- | --- |
| count\_max\_value : STD\_LOGIC\_VECTOR (3 downto 0) := "0000"); | Maximalwert Zähler |

# Beschreibung

Im Universalzähler counter kann der maximale Wert bis zu welchem gezählt werden soll durch ein generic gesetzt werden. Im counter\_3\_9 wird anhand der Zehnerstelle, also dem Wert des nachfolgenden Zählers, entschieden ob bis 3 oder 9 gezählt wird. So wird zwischen 19Uhr und 23Uhr unterschieden. Ansonsten sind die Modul wie Standardzähler. Die exakte Realisierung kann dem Code entnommen werden welcher mit der Dokumentation mitgeliefert wird.

# Testbench

In der Testbench wird das korrekte laden sowie das korrekte zählen überprüft. Dazu werden die Eingangtakte simuliert und Daten an den Load-Eingängen angelet welche nach dem Reset geladen werden. Danach werden die ce sowie ci Eingange angeregt.



Die Simulation zeigt den Fall wo eine 5 geladen wird. Danach werden ce und ci abgeschaltet um zu testen, ob der Zähler aufhört zu zählen. Da ten\_count auf 1 ist gibt der Zähler einen Übertrag an co aus beim Zählerwert 9.