Tutorial für prMagic v1.0

Maximilian Diedrich

21. September 2017

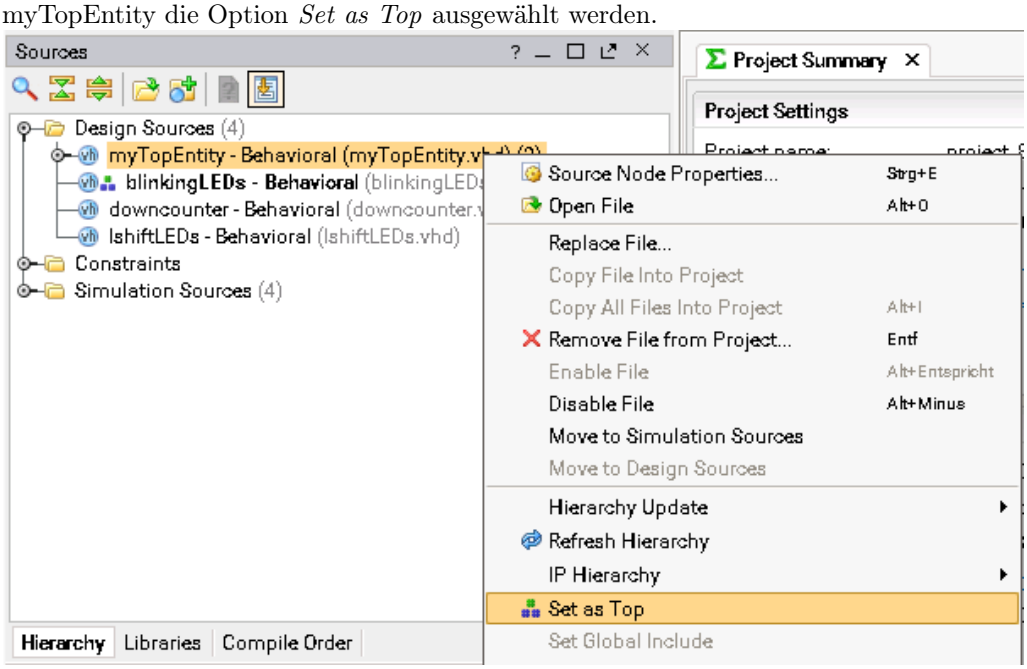
1 Vorraussetzungen

- Vivado 2016.3 oder neuer
- Vivado *partial reconfiguration* Lizenz
- prMagic v1.0 https://github.com/diedricm/prMagic/releases
- prMagic Tutorial Code https://github.com/diedricm/prMagicTutorial

Ein Projekt mit dem Ergebnis dieses Projekts befindet sich im Ordner **project**. Der für dieses Tutorial zu verwendende Code ist im Ordner **src**.

2 Arbeitsschritte

1. Lege ein neues Vivado-RTL-Projekt an.
2. Importiere den VHDL Beispiel Code. Die Dateien sind:
   - **src/blinkingLEDs.vhd**
   - **src/downcounter.vhd**
   - **src/lshiftLEDs.vhd**
   - **src/myTopEntity.vhd**
   - **src/rshiftLEDs.vhd**
   - **src/upcounter.vhd**
3. Es müssen keine IP importiert werden.
4. Importiere main.xdc für die IO-*constraints*.
5. Wähle für dieses Tutorial das Zedboard Zynq Evaluation and Development Kit als Zielsystem aus.
6. Drücke *Finish*, um den *wizard* zu beenden.
7. Stelle sicher, dass myTopEntity im *hierarchy-view* die *top-level-entity* ist. Wenn das nicht der Fall ist, dann kann, wie in der Darstellung zu sehen ist, mit einem Rechtsklick auf myTopEntity die Option *Set as Top* ausgewählt werden.



Dies kann auch mit dem TCL-Kommando:

```
set_property TOP myTopEntity [get_filesets sources_1]
```

erreicht werden.

8. Starte die RTL-Analyse, um den Code auf syntaktische Fehler zu testen.
9. Definiere als nächstes die dynamischen Module. Füge dafür direkt über der *component instantiation* **leddriver** in der *architecture* Behavioral der *entity* myTopEntity die folgenden Anweisungen ein:

```
--prmodule (rshiftLEDs);
--prmodule (blinkingLEDs);
```

Der Code soll danach so aussehen:

```
--prmodule (rshiftLEDs);
--prmodule (blinkingLEDs);
leddriver: rshiftLEDs
port map(
        trigger => itrigger,
        leds => leds
);
```

Dadurch wird eine Partition **leddriver** mit den Modulen lshiftLEDs, rshiftLEDs und blinkingLEDs erzeugt.

10. Füge analog dazu das dynamische Modul downcounter zu der *component instantiation* triggercoutnter hinzu

11. Stelle sicher, dass die Dateien prMagic.tcl und vhdlExtract.jar im momentanen Arbeitsverzeichnis oder im Systempfad sind. Lade das prMagic Script mit:

    ```
    source prMagic.tcl
    ```

12. Führe prMagic zur automatischen Konfiguration des Projektes auf. Verwende hierfür den eben geladenen Befehl:

    prm_main

13. Starte die Synthese.

14. Wähle **open synthesized design** und markiere die blackboxes leddriver und triggercounter je mit einem Rechtsklick und wähle **Floorplan > draw pBlock**. Zeichne einen pBlock, der:

    (a) Mindestens 20% mehr Ressourcen als das größte zu ihm gehörende Modul benötigt hat.

    (b) Für 7Series FPGAs: Vertikal durch die Grenzen der Taktregion beschränkt wird. Für Ultrascale FPGAs gilt diese Beschränkung nicht.

    (c) Horizontal keine Interconnect-Spalte trennt.

    Setze für beide pBlocks den Parameter RESET_AFTER_RECONFIG auf 1 und den Parameter SNAPPING_MODE auf ON.

15. Teste mit den Design Rule Checks durch das Klicken auf **Tools > Report > Report DRC** das Projekt auf Fehler. Wähle nur die Partial Reconfiguration Tests aus und starte die Tests.

16. Speicher die *constraints*.

17. Starte die Implementierung.

18. Starte die Generierung der Bitstreams.

19. Die fertigen bitfiles befinden sich nun unter:

    - **<projectname>.runs/prm_child_3_impl_1/triggercounter_greybox_partial.bit**
    - **<projectname>.runs/prm_child_3_impl_1/leddriver_blinkingLEDs_partial.bit**
    - **<projectname>.runs/prm_child_2_impl_1/triggercounter_downcounter_partial.bit**
    - **<projectname>.runs/prm_child_2_impl_1/leddriver_lshiftLEDs_partial.bit**
    - **<projectname>.runs/impl_1/leddriver_rshiftLEDs_partial.bit**
    - **<projectname>.runs/impl_1/triggercounter_upcounter_partial.bit**
    - **<projectname>.runs/impl_1/myTopEntity.bit**