Practicum 3:
Communicerende automaten

Caro Meysmans
Caro.Meysmans@UGent.be

Lukas De Loose
Lukas.DeLoose@UGent.be

Garben Tanghe
Garben.Tanghe@UGent.be

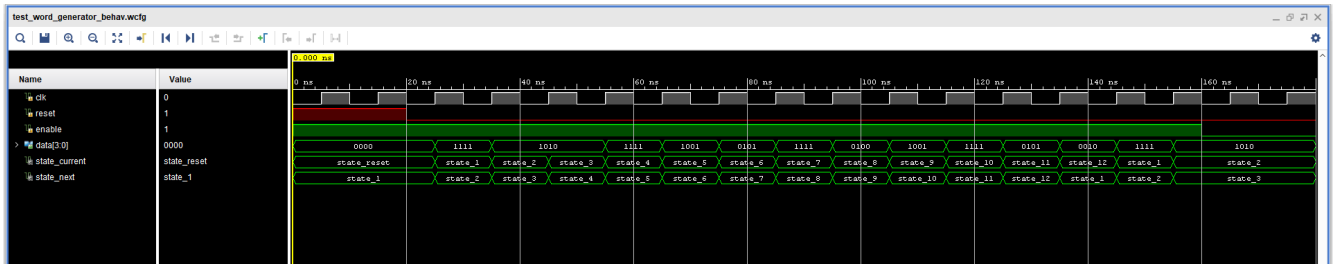
10 december 2017

Inhoudsopgave

Lijst van figuren

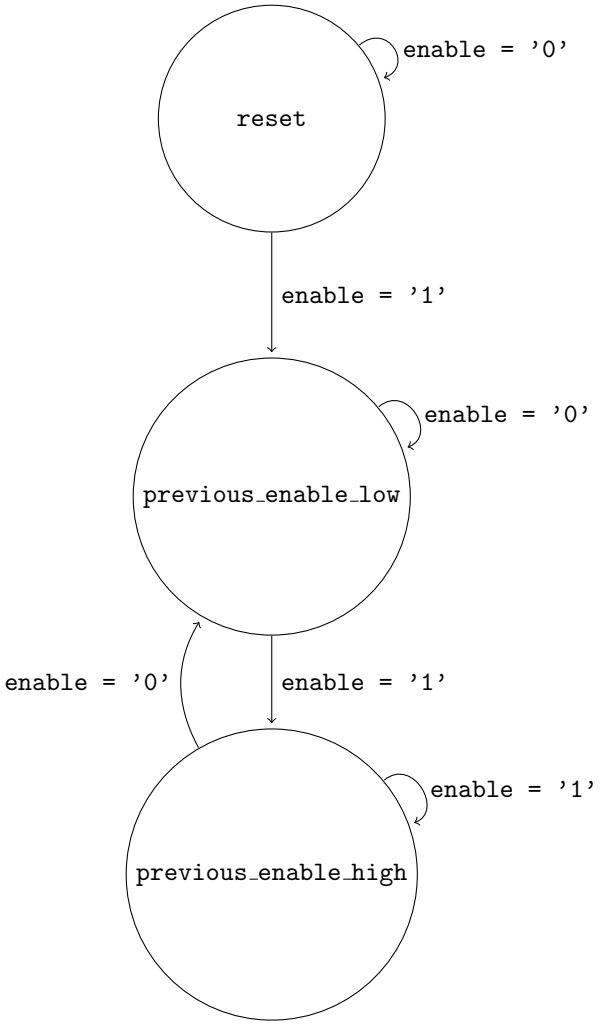
1 De woordgenerator

De woordgenerator implementeren we als een synchrone automaat met 13 toestanden. De resettoestand zet de uitgangsbits op `"0000"`. De reset is asynchroon. De overige 12 toestanden zetten de gewenste sequentie op de uitgangsbits. We bemerken dat wanneer de `enable`-ingang laag staat, er geen nieuw woord naar buiten wordt gebracht.



Figuur 1: `word_generator` gedragssimulatie

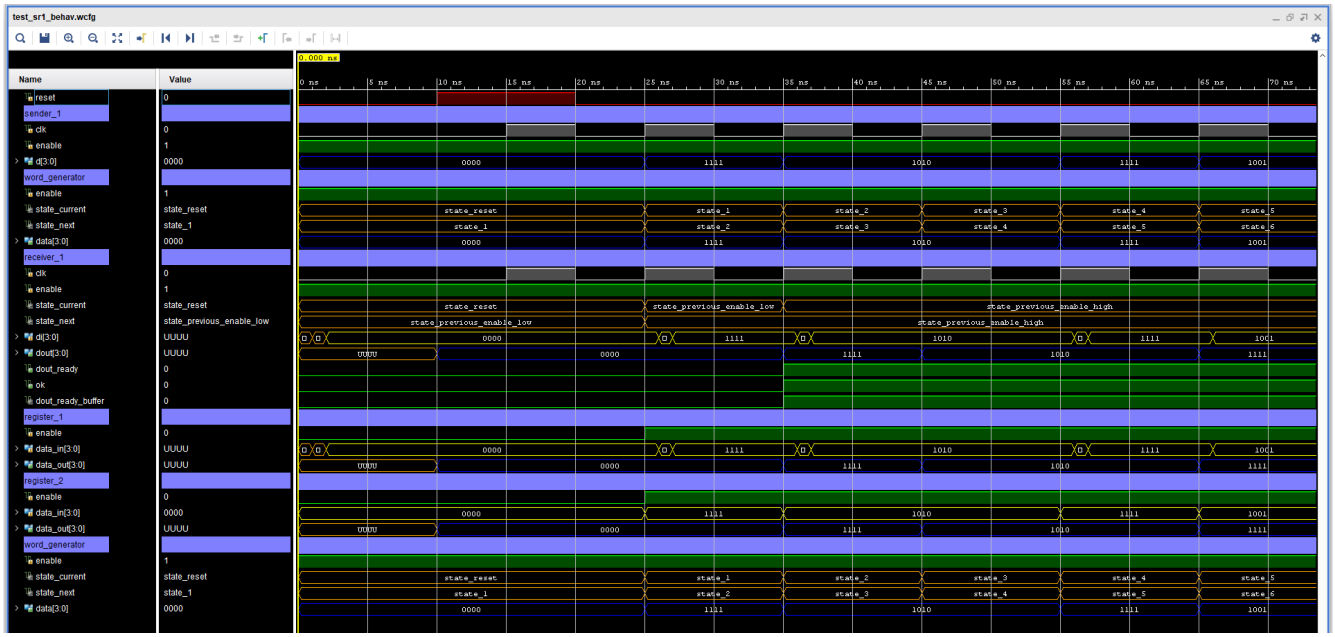
2 De eerste zender en ontvanger



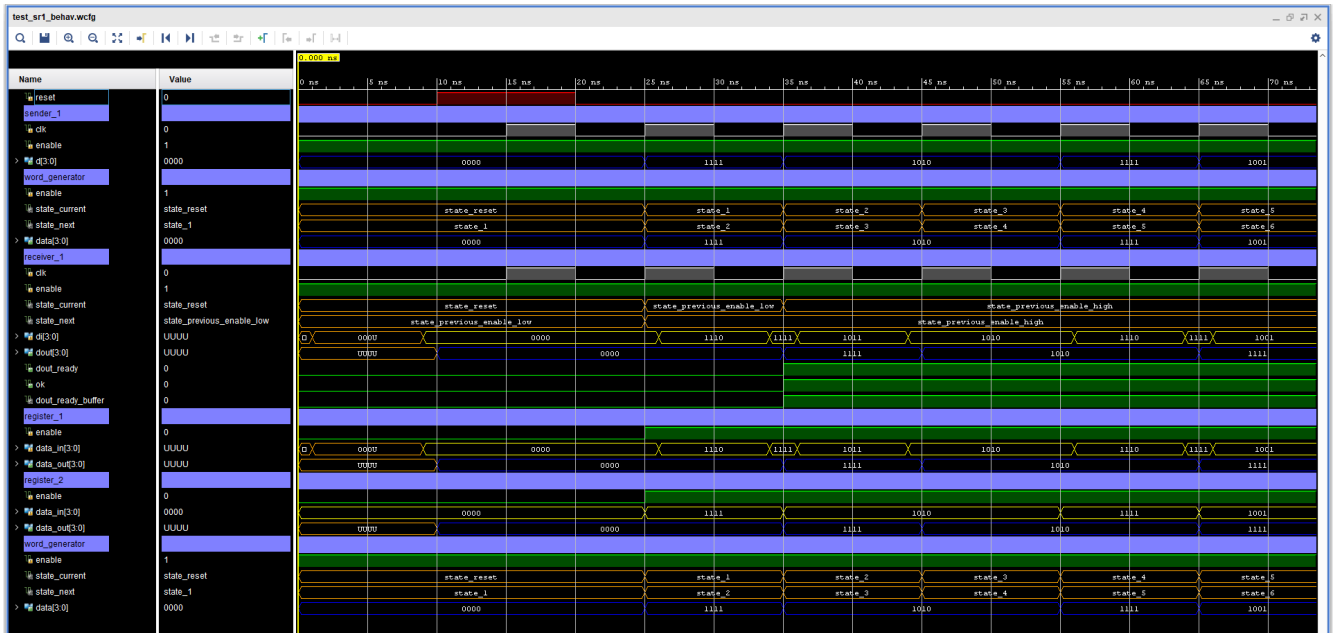
Figuur 2: `r1` toestandstransitiediagram

Wanneer verwacht je dat het mis zal lopen? Toon aan met simulaties dat je verwachtingen kloppen. Daar de klokperiodes van zender en ontvanger gelijk zijn, verwachten we dat het mis zal lopen als `delay2` groter wordt dan de klokperiode van de ontvanger. In deze situatie is het woord aan de ingang van de ontvanger niet op tijd stabiel. De volgende gedragssimulaties staven onze verwachtingen.

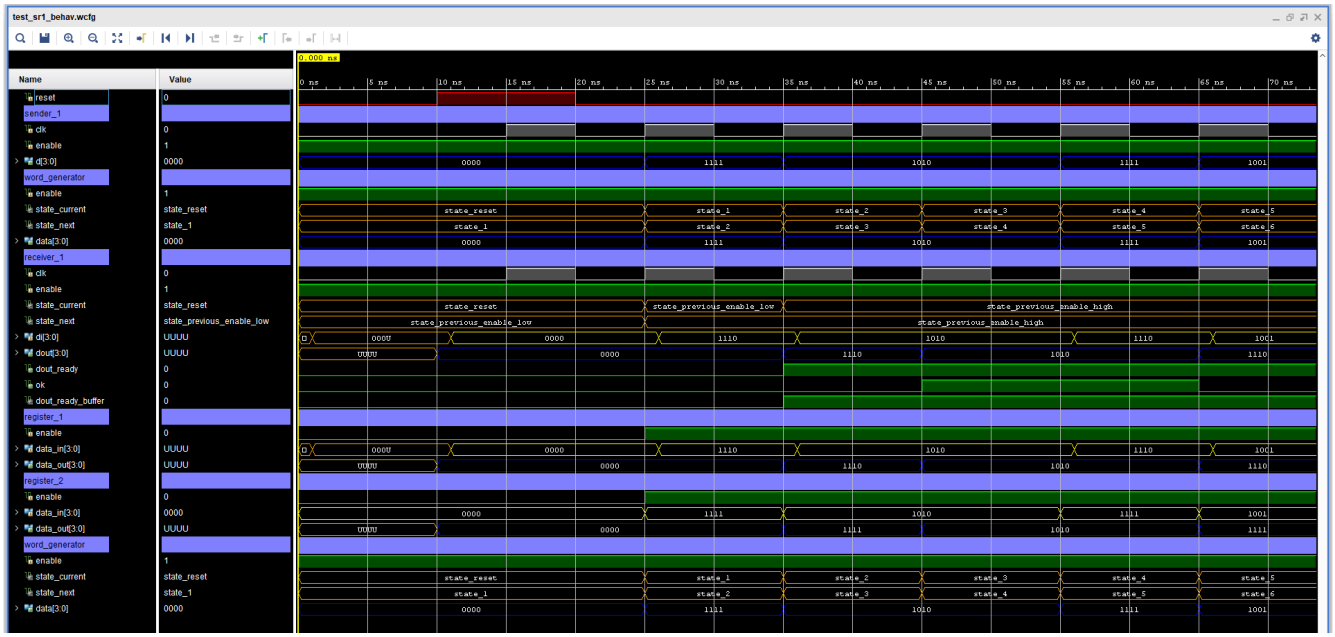
Nu laten we de zender op een kloksignaal van 10 ns lopen. Indien we vervolgens de ontvanger op een kloksignaal van 7 ns en 13 ns laten lopen, bemerken we dat de automaten niet meer succesvol communiceren. Het verschil in klokperiodes zorgt dat beiden woordgeneratoren niet synchroon lopen. In de eerste situatie zal de ontvanger meermaals hetzelfde woord inlezen, aangezien de zender niet tijdig een nieuw woord kan genereren. In de tweede situatie zal de ontvanger woorden overslaan, aangezien deze niet tijdig klaar is om een nieuw woord in te lezen.



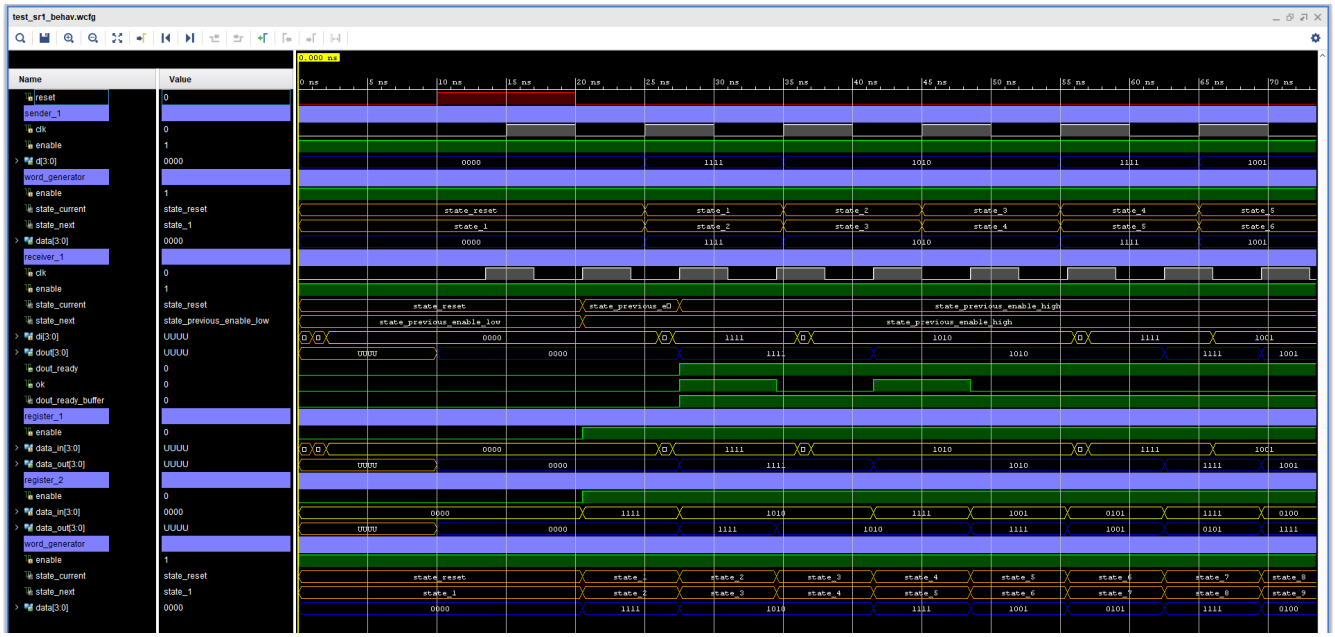
Figuur 3: `s_1`, `r_1` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 10\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 2\,\text{ns}$



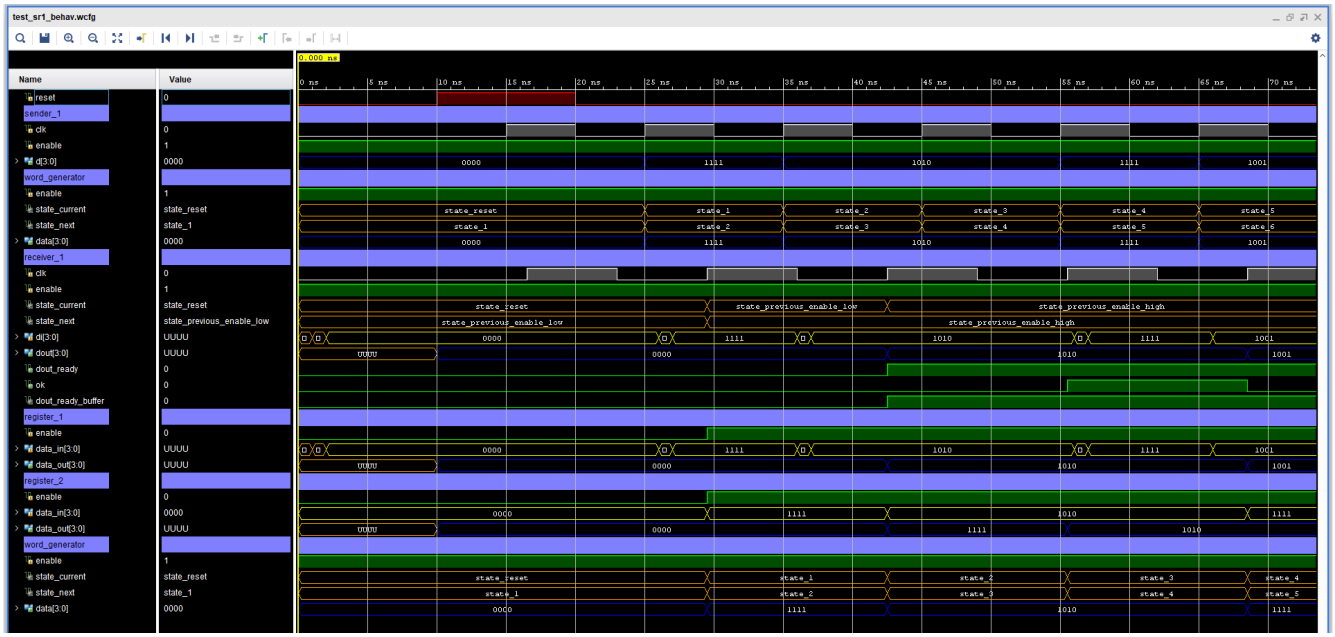
Figuur 4: `s_1`, `r_1` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 10\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 9\,\text{ns}$



Figuur 5: `s_1`, `r_1` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 10\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 11\,\text{ns}$

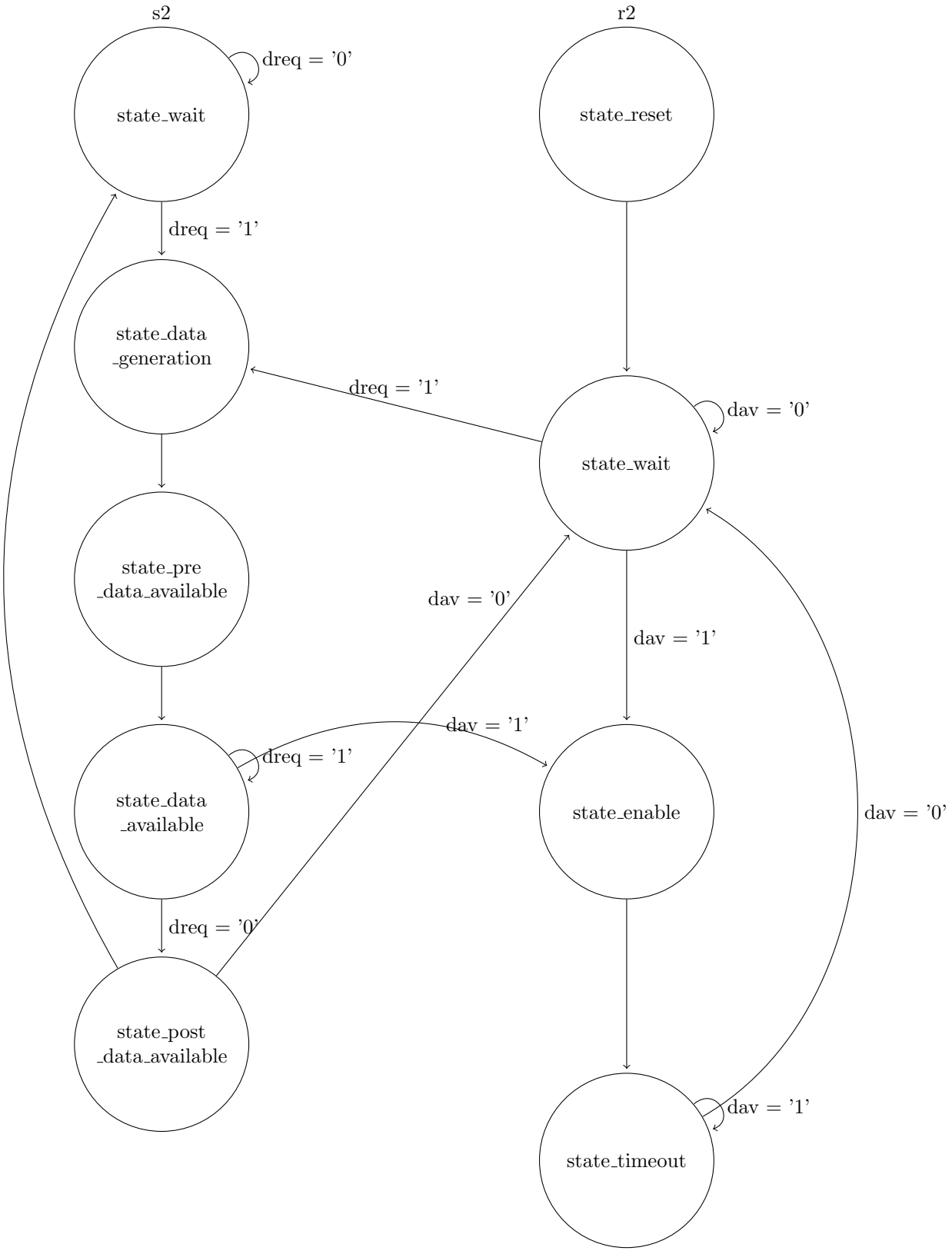


Figuur 6: `s_1`, `r_1` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 7\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 2\,\text{ns}$



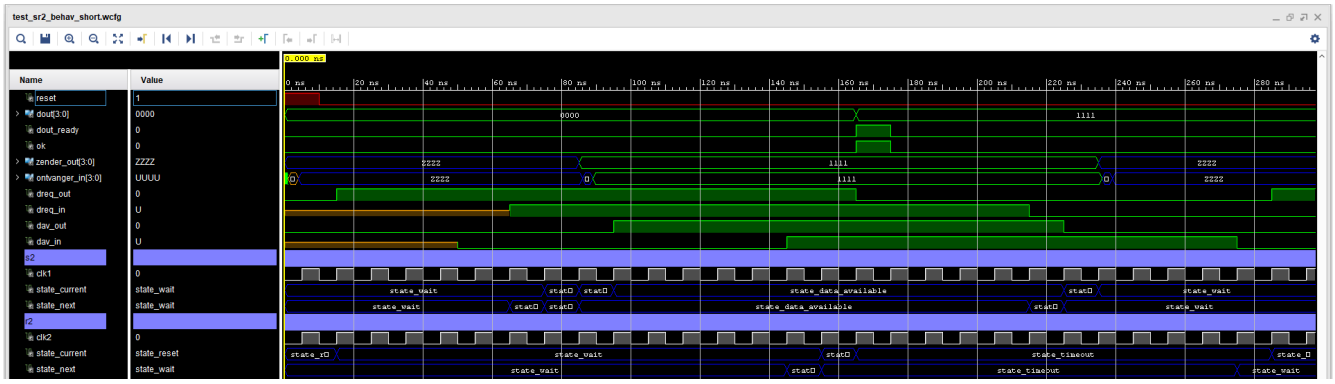
Figuur 7: `s_1`, `r_1` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 13\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 2\,\text{ns}$

3 De tweede zender en ontvanger

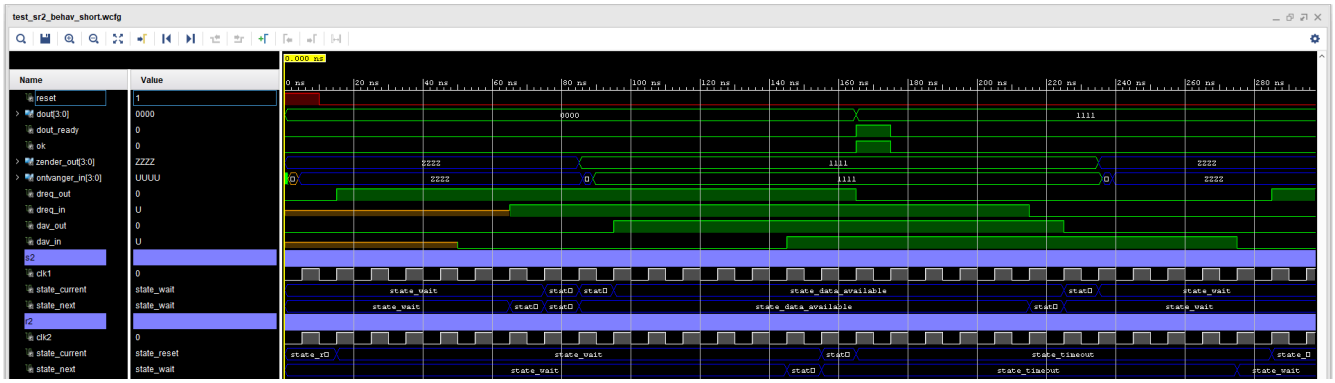


Figuur 8: `r2 en s2` toestandstransitiediagram

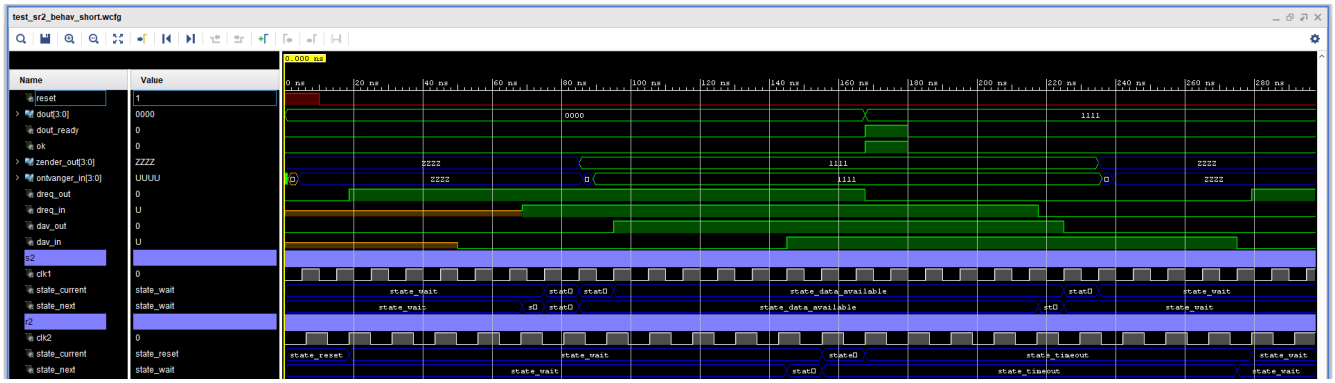
Wordt in dit protocol een veronderstelling gemaakt over de relatieve vertragingen van de signalen tussen beide automaten? Leg uit.
We veronderstellen dat de vertraging op controlesignalen groter is dan de vertraging op de datasignalen. Indien dit niet het geval is, zal de zender immers aangeven dat zijn uitgangsbits stabiel zijn, terwijl de ingangsbits van de ontvanger nog niet stabiel zijn. Een voorbeeld van dit gedrag vinden we in figuur 13. We bemerken dat zelfs bij zeer grote vertragingen op het `DAV`-signaal de communicatie tussen de automaten succesvol blijft. Dit wordt gegarandeerd door de 4-way handshake.



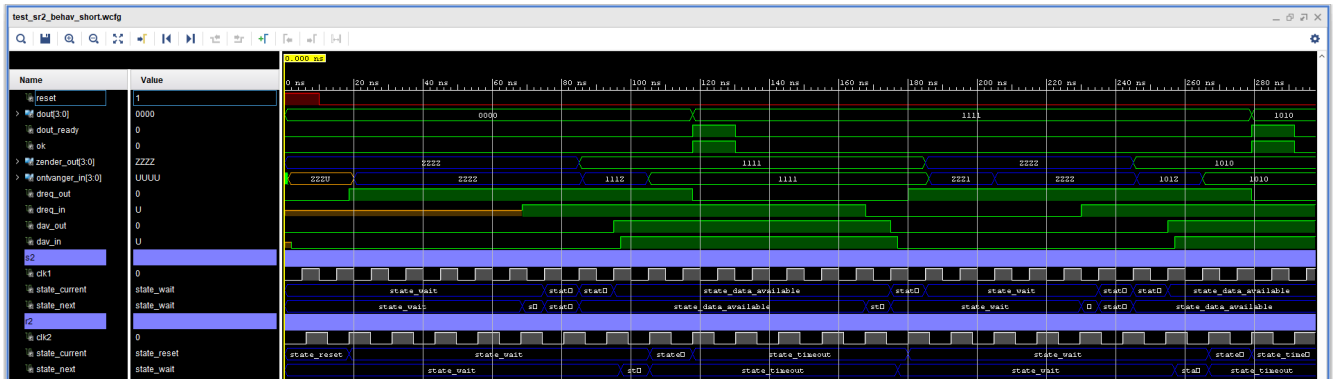
Figuur 9: `s_2`, `r_2` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 10\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 2\,\text{ns}, \Delta_3 = 1\,\text{ns}, \Delta_4 = 2\,\text{ns}$



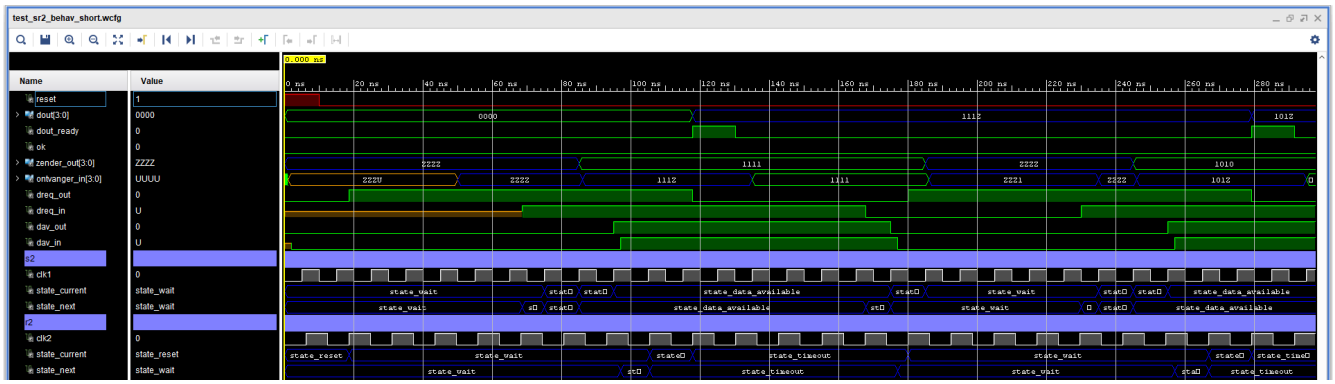
Figuur 10: `s_2`, `r_2` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 10\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 4\,\text{ns}, \Delta_3 = 50\,\text{ns}, \Delta_4 = 50\,\text{ns}$



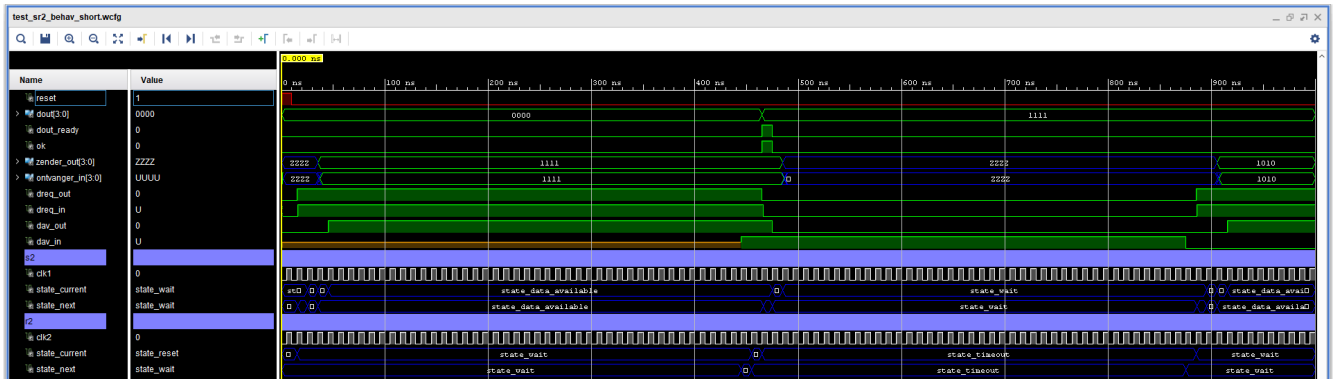
Figuur 11: `s_2`, `r_2` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 12.4163\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 4\,\text{ns}, \Delta_3 = 50\,\text{ns}, \Delta_4 = 50\,\text{ns}$



Figuur 12: `s_2`, `r_2` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 12.4163\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 20\,\text{ns}, \Delta_3 = 50\,\text{ns}, \Delta_4 = 2\,\text{ns}$



Figuur 13: `s_2`, `r_2` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 12.4163\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 50\,\text{ns}, \Delta_3 = 50\,\text{ns}, \Delta_4 = 2\,\text{ns}$



Figuur 14: `s_2`, `r_2` gedragssimulatie bij $T_1 = 10\,\text{ns}, T_2 = 10\,\text{ns}, \Delta_1 = 1\,\text{ns}, \Delta_2 = 2\,\text{ns}, \Delta_3 = 1\,\text{ns}, \Delta_4 = 400\,\text{ns}$

4 Synchronisatieflipflop

Bedenk voor welke toestandsassignatie dit bij jullie ontvanger-automaat voor problemen kan zorgen. Leg duidelijke uit wat er fout loopt aan de hand van een schets van simulatiegolfvormen (ontvangerklok, DAV en de individuele toestandsbits van de ontvangerautomaat) waarin de volgorde van de gebeurtenissen en de vertragingen zo zijn gekozen dat het fout loopt. Toon ook aan voor dezelfde situatie hoe een synchronisatieflipflop het probleem oplost.

In de ontvanger kunnen er zich 2 kritische races voordoen bij een toestandstransitie. Bij een overgang van enable naar reset en van time out naar wait moeten er 2 toestandsbits aangepast worden, de volgorde waarin dit gebeurd is hierbij van belang aangezien de mogelijke tussentoestanden telkens stabiel zijn. Door te synchroniseren met een synchronisatieflipflop (Hamming-afstand 1), zullen de toestandsbits al meer dan 1 klokcyclus stabiel zijn.

5 Besluit

Dit protocol biedt nog altijd geen garantie op correcte werking in alle gavallen! Vertragingsverschillen tussen controlesignalen en de datalijnen zijn hier de boosdoener. Als deze delays meer dan een klokperiode (van de zender) verschillen, is het mogelijk dat dav stabiliteit van de data aangeeft, voor alle datatransities stabiel zijn aan de ingangen van de ontvanger. Correcte werking vereist dus dat de vertragingsverschillen tussen data- en controleverbindingen zo klein mogelijk zijn en dat er een bovengrens gekend is.