

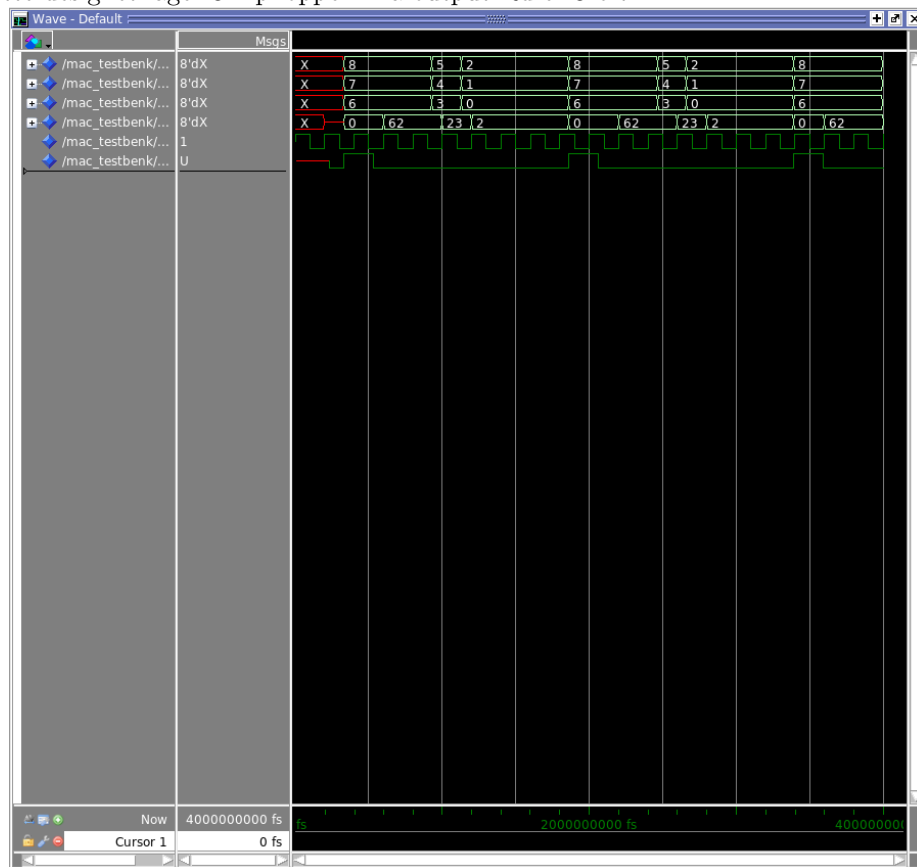
IN2060 - Oblig 1

Georg Mageshaugen - georgmag

26. September 2019

Del 1

Dette designet lager 8 flipfloppe. Da output Rd er 8 bit.



Del 2

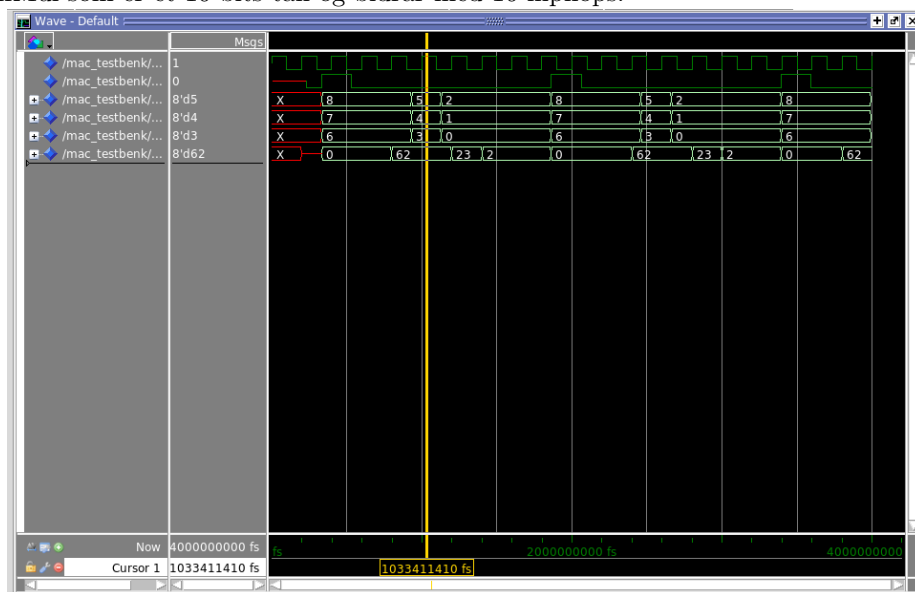
Forventet samme utregning, da tallene som ble regnet ut var de samme. Forutså ikke at de skulle bli forskøvet med en klokkeflanke, men det gir mening ettersom man mellomlagrer signalet før det blir regnet ut.

Fordelen med å innføre en pipeline er at man plukker ut signalet og lagrer det, når man i mye større grad kan forutse hva signalet er. Da trenger man i mye mindre grad å ta hensyn til forskjellige propageringsforskjellene mellom de ulike portene.

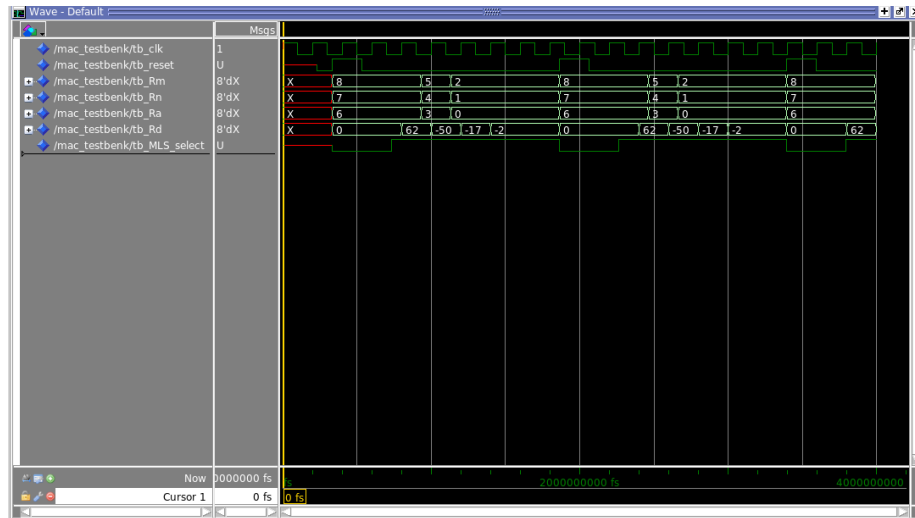
Dette designet har nå $8 + 8 + 16 = 32$ flipfloppe.

Rd og buffAdd som er 8-bits tall, bidrar med 8 flpsflops hver seg.

buffMul som er et 16-bits tall og bidrar med 16 flpsflops.



Del 3



Del 4

Den nye koden med det siste nivået med pipeline vil generere 66 flipfloppe:

buffProd, buffSum og buffSub er alle 16-bit signaler og bidrar med $3 * 16 = 48$ flipfloppe.

buffRa og Rd er 8-bit signaler og bidrar med $8 * 2 = 16$ flipfloppe.

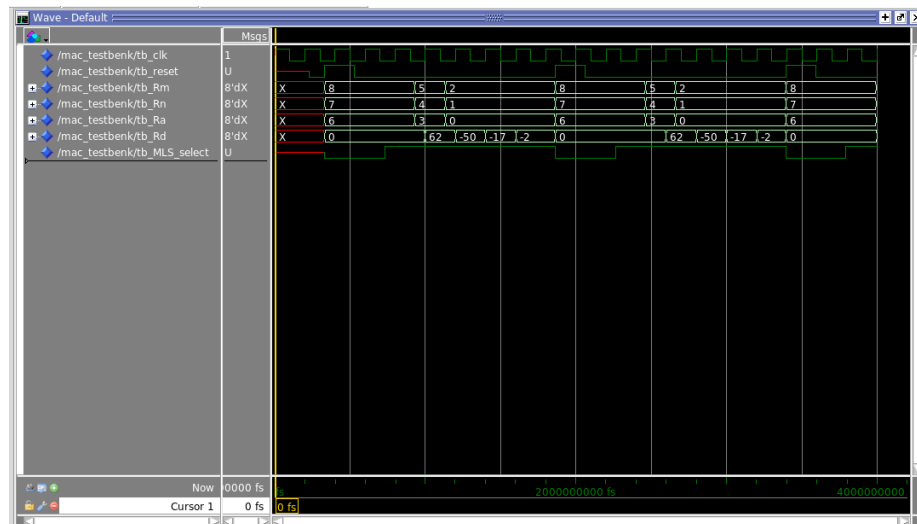
buffMLS og buffMLS2 er begge 1-bit signaler og bidrar $2 * 1$ flipfloppe.

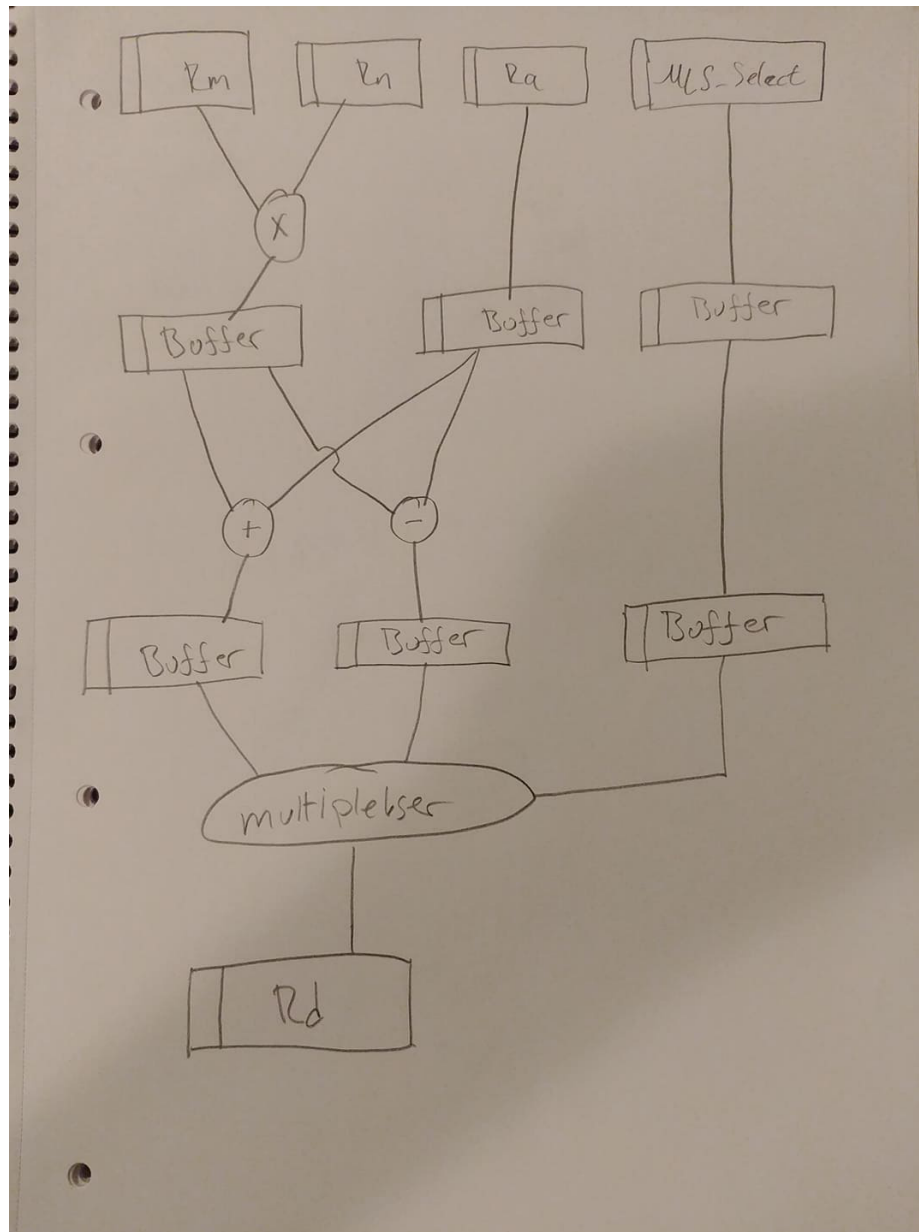
Altså $3*16 + 2*8 + 2*1 = 66$.

Det vil gå to klokkesykler fra input registerene er satt til resultatet kan leses fra Rd.

Hvis man ikke innfører det siste nivået med pipelining er det mulig at man kan få noe feil verdier en liten periode ettersom propageringsforsinkelen til sum og sub trolig ikke er helt lik. Hvis man er avhengig av å ha helt stabile signaler ut til Rd vil jeg si det er hensiktsmessig å innføre dette siste nivået.

Dersom forskjellen i propageringsforsinkelse mellom sum og sub er liten nok til at man tåler ustabilitet i minst like stor periode som denne forskjellen i vil det ikke være hensiktsmessig med dette siste nivået med pipelining, og man kan spare seg for noen flipfloppe og plass på kretskortet.





Skjema etter siste nivå med pipeline