TurboR op 40 MHz

Dennis Koller
MSX Computer & Club Magazine nummer 86 - november/december 1996

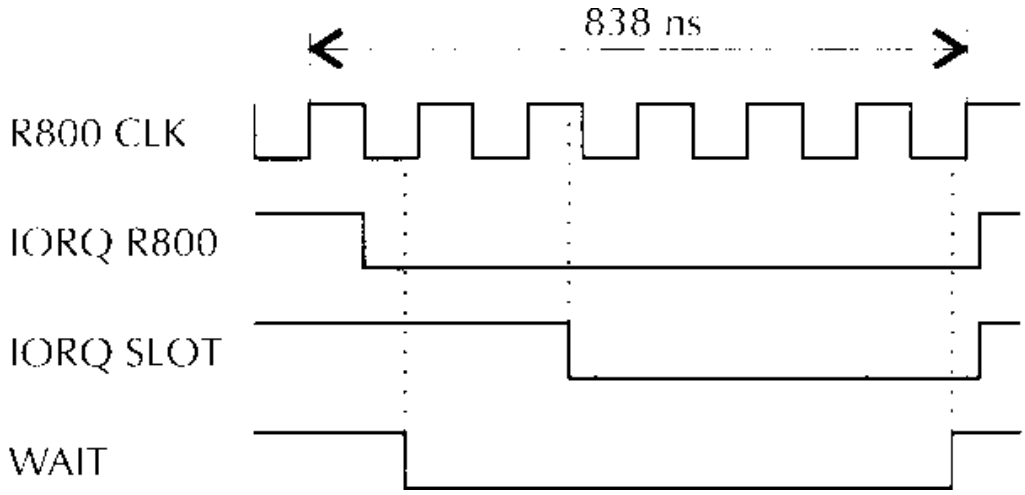
Scanned, ocr'ed and converted to PDF by HansO, 2001

Het opvoeren van de MSX TurboR blijkt een snelheidswinst te geven. Probleem is dat de VDP het dan eigenlijk niet meer aan kan. Ik heb dat opgelost door de VDP ook op te voeren en de monitor aan te pakken. De problemen met de bios scroll zijn in R800 mode voorbij en sinds die aanpassing draait mijn MSX TurboR standaard op 40 MHz.

Het opvoeren door het 28 MHz kristal te vervangen door een 40 MHz oscillator van een VGA kaart, geeft een redelijke snelheidswinst. Ook de VDP moest toen worden aangepast en daarom heb ik het 21 MHz kristal eruit gebroken en er een van 27.64 MHz ingezet. Een aardige ingreep, maar die had tot gevolg dat mijn monitor erg raar ging doen, maar dat maakte mij niet uit, want ik verbouwde dan de monitor ook maar. Het vervelende is eigenlijk dat in Z80 mode dit probleem niet over is, de Z80 is met OUT's kennelijk sneller. Ik heb dit opgelost door in de AUTOEXEC.BAS een return op adres &H0180 in de bios te zetten, zodat de computer altijd op R800 dram staat. Zie het programmadeel hieronder.
Maar dit betekent natuurlijk wel dat de R800 overdreven wordt vertraagd, als die Z80 hem inhaalt. Dus, computer openschroeven...

Wait

Om te beginnen heb ik in WBASS een programmaatje geschreven waarin alleen een instructie IN steeds werd herhaald. Nu blijkt dat als dit programma op R800 draait, de IORQ niet minder lang actief is dan bij Z80, maar de pulsen komen sneller na elkaar. Het wait-signaal afkomstig van de SI 990 doet het volgende:

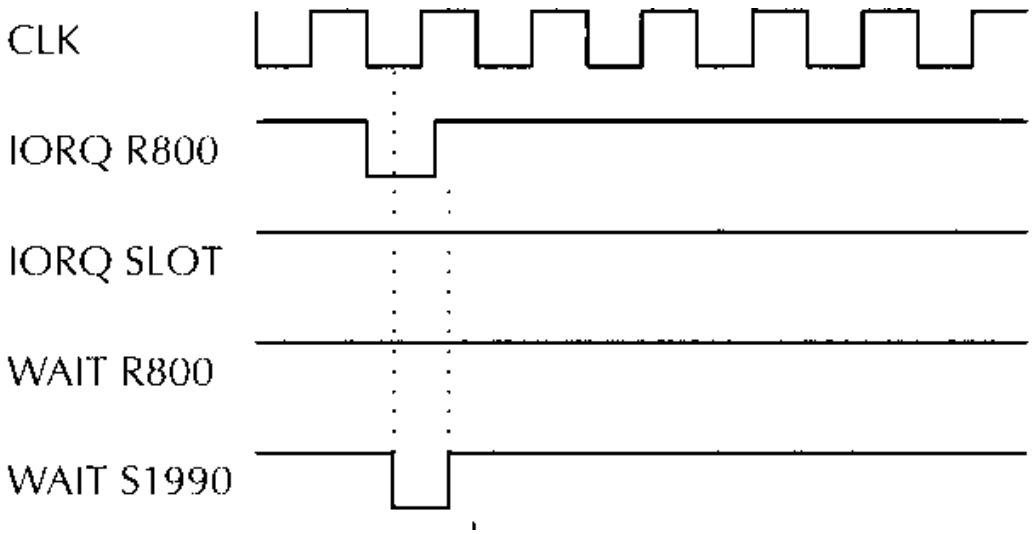


De S1990 laat de R800 bij elke I/O-poort minstens 698 ns, oftewel 5 cycli, wachten. Op 28 MHz is de wachttijd voor de VDP-poorten 98, 99, 9A en 9B 5.8 ms, of 42

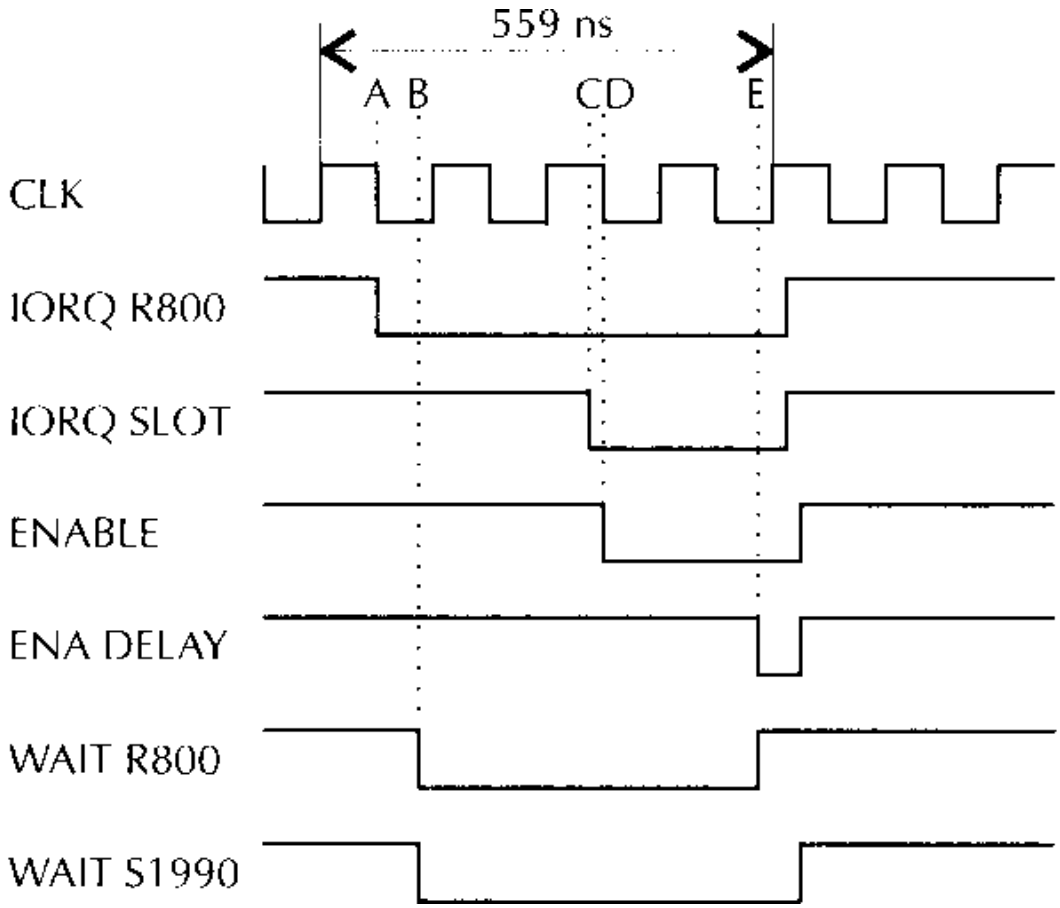
cycli. Op 40 MHz is dat nog te kort. Voor de printerpoorten 90, 91 en 93 is de wachttijd 1.4 ms, oftewel 10 cycli. Alle andere poorten kennen een wait van 0.7 ms, oftewel 5 cycli.
Tijd om die wait-lijn door te krassen. Hij start niet op: olé. Dan maar proberen om bij sommige I/O-poorten de wait te blokkeren: wait R800 = wait S1990 + A7 * IORQ
Dit leidt ertoe dat de wait bij elke I/O onder 128 wordt onderbroken.
Hij start op, dus WBASS en de oscil-loscoop erbij. Met het volgende programma kunnen een I/O-aanroep onder en boven de 128 met elkaar vergeleken worden:

```
        ORG &HC000
        DI
        IN A,(&HAA) AND 240 OR 8
        OUT (&HAA),A
LUS:    IN A,(&HOO)
        IN A,(&HA9)
        RRA
        JR C,LUS
        EI
        RET
```

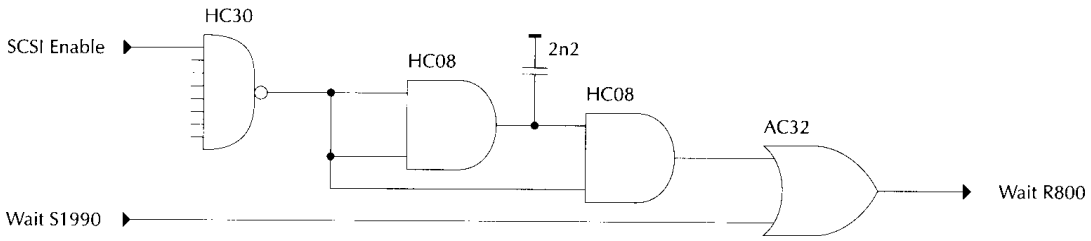
IN A,(&HOO) geeft het volgende resultaat:



Ik las een tekstbestand van Henrik Gilvad over de scsi van Gouda; die zou waarschijnlijk de snelheid wel aankunnen. Door te meten waar de pootjes van de PEEL heen gaan, bleef alleen pin 14 over om de enable van de interface chip te zijn. Dit signaal wordt gebruikt om via een vertraging het wait-signaal te blokkeren. Zonder vertraging is het enable-signaal te kort actief, want de S1990 vertraagt ook de databus van het slot naar de R800.
Om verder te kunnen meten, heb ik het eerder genoemde programma weer gebruikt, maar nu niet met IN A,(&HOO) maar met IN A,(&H34), waar de scsi zit. Dit gaf het volgende resultaat:



Op het moment dat de R800 een I/O-aanroep doet (A), genereert de S1990 iets later het wait-signaal (B). Een klokcyclus later maakt de S1990 IORQ van de externe bus actief (C), waarop na 25 ns scsi enable volgt (D). Het scsi enable-signaal wordt iets vertraagd, om er zeker van te zijn dat de data goed overkomt en als het scsi enable-signaal door de vertraging heen is, wordt de wait geblokkeerd (E). Doordat de R800 niet meer wordt opgehouden, maakt hij IORQ weer inactief en gaat verder. Gelukkig maakt de S1990 het wait-signaal daardoor ook direct weer inactief, waardoor de R800 niet in de knoei kan komen met bijvoorbeeld een INIR instructie. Dit geeft per I/O-cyclus al 279 ns winst. Deze I/O-cyclus duurt tweederde van de tijd van een normale cyclus.



Figuur 1: nowait

Figuur l laat het schema nowait zien. De condensator tussen de twee AND-poorten bepaalt de vertraging. De schakeling kan zowel voor I/O-poorten — scsi — als voor

geheugen worden gebruikt met behulp van het SLTSL-signaal. Het schema van nowait1 in figuur 2 doet in wezen hetzelfde, maar er zijn veel meer poorten voor nodig. Het is alleen maar handig voor devices die zo snel zijn, dat ze geen vertraging nodig hebben, want die kunnen bij deze schakeling gemakkelijk naar keuze weggelaten worden door het signaal direct naar de NAND te leiden in plaats van via de OR-poorten. Maar in 99% van de gevallen is de vertraging toch nodig en dan is nowait veel voordeliger. Voor alle schakelingen geldt dat de niet gebruikte ingangen van de NAND aan Vcc moeten worden gelegd. Zorg er verder voor dat alle IC's worden ontkoppeld met een condensator van 100 nF over de voeding. Ik had het eerst ook niet gedaan en na een kwartier tot drie uur gebeurden er rare dingen. En dan moet je de fout maar vinden...
De vertraging is zo ingesteld, dat de periode dat IORQ van de externe bus actief is, geen vier maar twee cycli bedraagt. Dit betekent dat de aanspreektijd van 558 ns naar 279 ns gaat op 28 MHz. Verder opvoeren naar één cyclus, door de vertraging van het enable-signaal eruit te halen, gaat meestal te snel in verband met de propagatievertraging van de S1990. Bij één cyclus is de aanspreektijd 139 ns. Zonder die vertraging is de tijd te klein voor de scsi. Met twee cycli is de aanspreektijd 279 ns en dat werkt prima.
Let wel, als IORQ van de externe bus vier cycli actief is, kost dit zes cycli van de R800. De R800 moet altijd twee cycli extra wachten door de vertraging van de S1990.

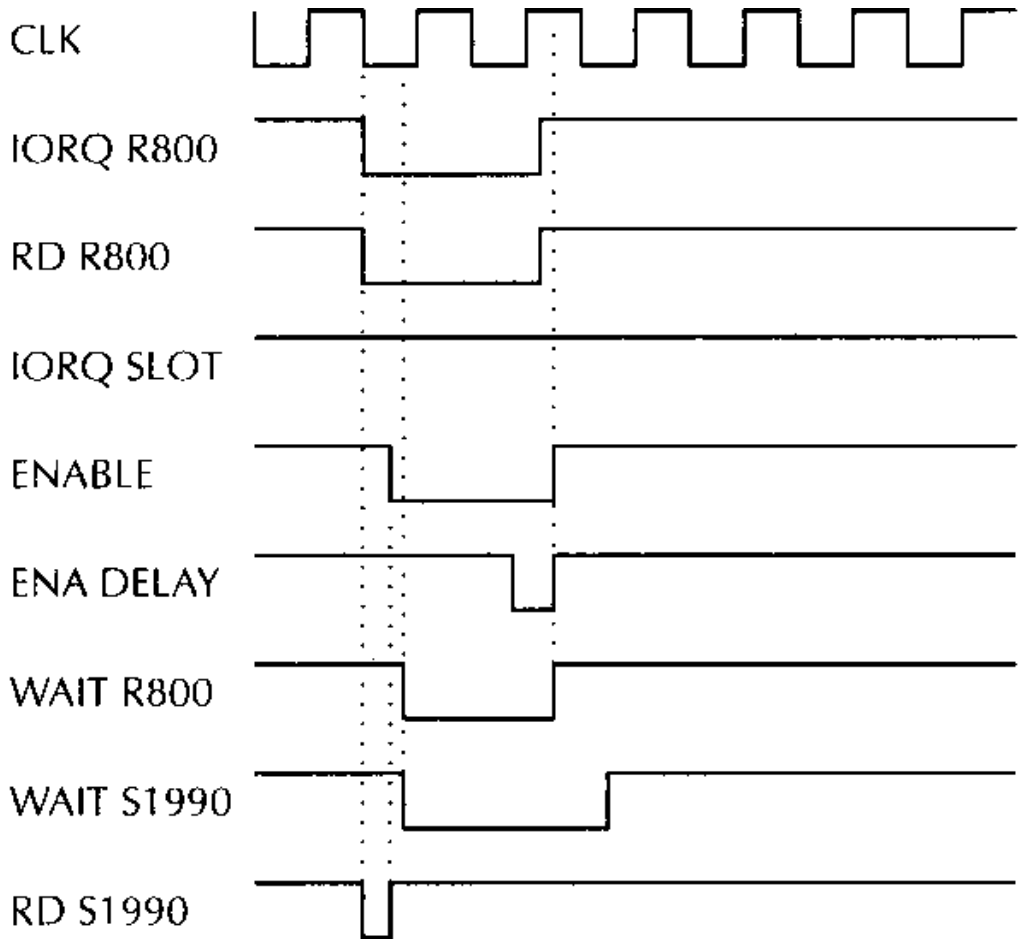
Geheugencartridge

Door de SLTSL-signalen te gebruiken, is ook een extern geheugen op te voeren. Niet kritisch voor rom's sneller dan 150 nS. Maar opvoeren van een ramcartridge heeft een grotere beperking, omdat het WRITE-signaal later komt dan alle andere signalen. Voor I/O-poorten geldt dit niet. Door het WR-signaal van het slot door te krassen en aan te sluiten op WR van de R800, is dat probleem opgelost.
Als de vertraging — de twee AND's voor de OR — eruit wordt gelaten, wordt het aanroepen van een extern slot nagenoeg twee keer zo snel. Er is alleen bijna niets dat die snelheid aan kan, maar met een cache ram — chip pin compatible met 62256/43256 of 27256 waarvan pin 27=WE — voor een pc (20 ns) werkt het. Ik heb een programma geschreven dat 2560 keer 16 kB kopieert; met een opgevoerd slot duurt dat 54 seconde en met een normaal extern slot 97 seconde.

Direct van R800

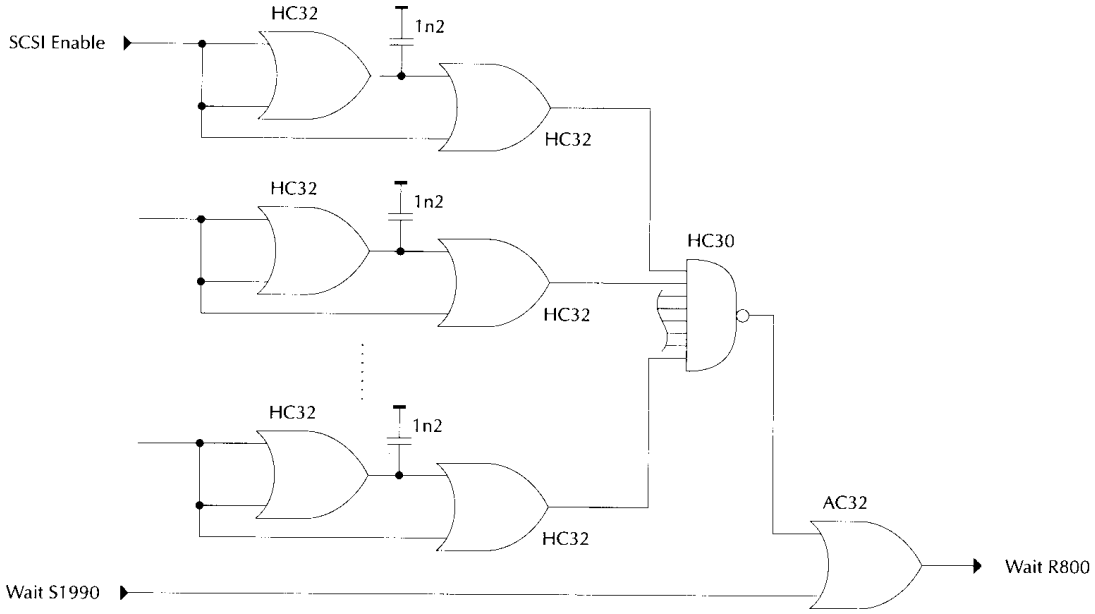
Het grote probleem in de MSX TurboR is de S1990. Die vertraagt veel te veel: hoe dan ook 279 ns. Bij het schrijven naar extern geheugen is het WR-signaal de beperkende factor, die wordt zo'n 350 ns vertraagd. Alles bij elkaar heeft de S1990 nog een propagatievertraging van circa 100 ns. Daarom werkt alleen een cache ram, met de WR direct van R800, terwijl de R800 420 ns de tijd geeft om data klaar te zetten. De rest van de tijd verdwijnt in de S1990.
De S1990 vertraagt hoofdzakelijk de address- en control-bus IORQ, MEMRQ, RD en WR van de R800. Deze vertraging kan worden omzeild door de address- en control-bus van de R800 direct op de scsi aan te sluiten, dus niet via de S1990. Gelukkig geeft de S1990 in Z80 mode de address- en control-bus van de Z80 volledig door aan de R800. Dus als de Z80 een aanroep doet, komen RD, WR, IORQ en MEMRQ gewoon bij de cartridge.
De vertraging van de databus is circa 40 ns, dus voor andere cartridges kan het nodig

zijn ook de databus direct van de R800 te nemen. Maar het is eenvoudiger de aanroeptijd een klokcyclus langer te maken. Als de databus van de R800 wel direct naar de cartridge gaat, moet voorkomen worden dat de S1990 ook data op de R800-bus gaat zetten bij het lezen van de betreffende cartridge. Dit kan door het RD-signaal van de R800 naar S1990 te blokkeren met behulp van het enable-signaal van de cartridge. Dit is nodig, omdat de S1990 bij alle externe adressen — anders dan dram — zich gedraagt als een buffer. Al staat er geen data op de ingang van een buffer, toch zal de buffer dit als enen zien en een keiharde logische een op de uitgang genereren. In dit geval zou er een botsing ontstaan tussen de S1990 en de cartridge. Voor de scsi zijn deze problemen niet van toepassing; deze kan gewoon op de standaard-databus aangesloten blijven.
Het voordeel van het gebruik zonder S1990, is dat de R800 veel minder lang op data hoeft te wachten, want iets aanroepen via de S1990 kost altijd drie klokcycli. Maar toch moet de R800 nog een cyclus wachten op de scsi, want anders duurt de aanroep maar 54 tot 70 ns en dat is toch te snel. Dit kan gewoon door het scsi enable-signaal aan te sluiten op de schakeling nowait; dan is de aanspreektijd anderhalve cyclus, oftewel 210 ns op 28 MHz en 163 ns op 36 MHz.
Een I/O-cyclus van de R800 met directe verbinding ziet er zo uit:



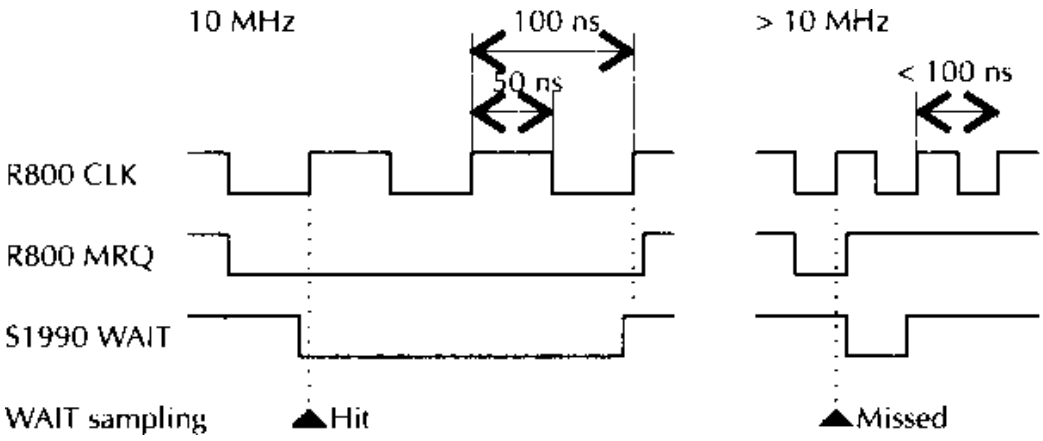
Dat ziet er beter uit, maar als de cartridge dit niet aan kan, is de gemakkelijkste methode de vertraging voor het blokkeren van het wait-signaal iets langer te maken. Want de databus van de R800 direct op de cartridge gaat niet zomaar. Als de databus zijn normale weg volgt, via de S1990, kan de Z80 de cartridge ook aanroepen, maar als de databus direct van de R800 komt niet. De S1990 vraagt in Z80 mode de

volledige bus vrij van de R800, maar geeft de databus van de Z80 niet door aan de R800. Wel RD, wat in dit geval dus een uitgang is van de S1990 en dat maakt het blokkeren ervan niet gemakkelijker. In nowait2, afgebeeld in figuur 3, wordt dit opgelost met een tristate buffer.

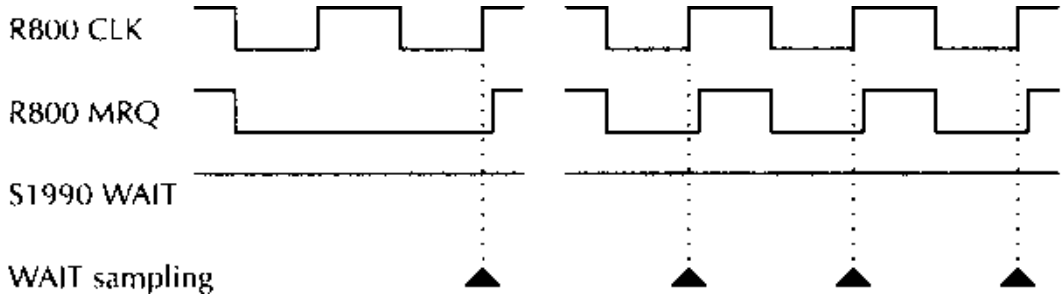


Figuur 2: nowait1

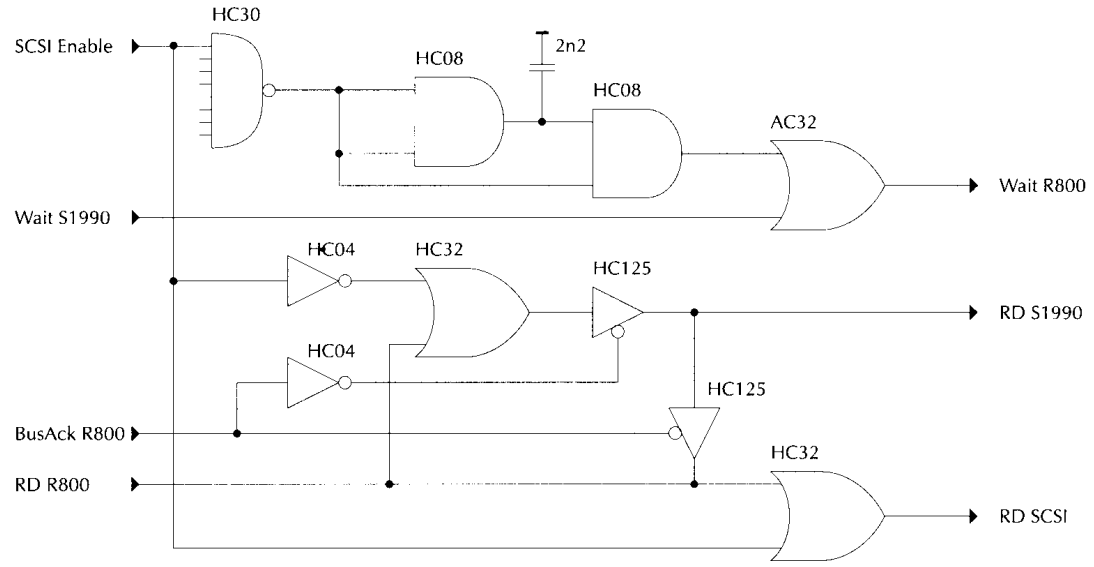
Het programma HDSPEED.COM gaf het volgende resultaat. Op 28 MHz: 528,51 kB/s; totale tijd 31 seconde. Op 33 MHz: 630,15 kB/s; totale tijd 26 seconde. Dit zijn snelheden van de laatste sector; uitschieters kwamen over 1.2 MB/s.
Het basicprogramma TESTSCSI.BAS geeft op 33 MHz het volgende aan. Bios 1: 216 kB/s. Dos: 744 kB/s. Er is een klein probleempje voor de 40 MHz-freaks, zoals ik. Door de — kleine — propagatievertraging van die OR-poort in het wait-signaal, komt het wait-signaal net te laat bij de R800, waardoor hij de aanroep beëindigt. Nadat IORQ of MEMRQ actief wordt, is er op 40 MHz hooguit 50 ns de tijd om het wait-signaal actief te maken.
40 MHz: CPU clock = 40:4 = 10 MHz
Aanroep van 'extern' geheugen, anders dan dram:



Aanroep van dram:
New MSB address: Within 256 bytes:



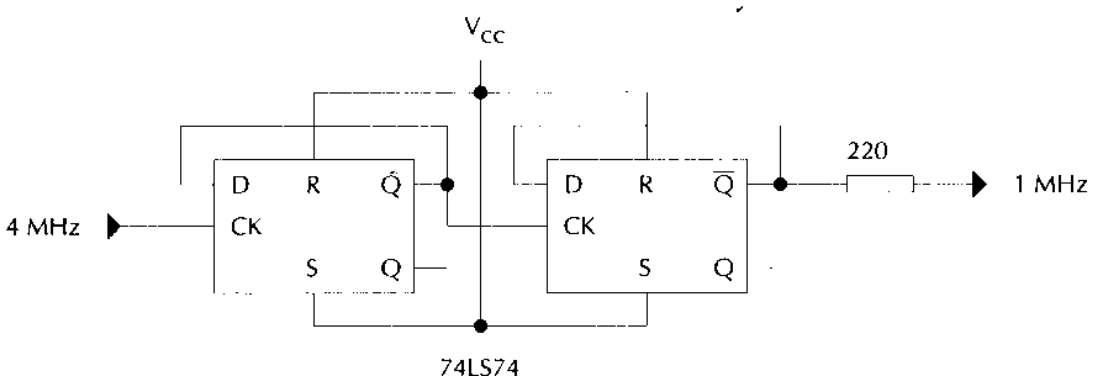
Als de R800 iets aanroept, doet hij dit op het moment dat de CLK laag is en verwacht een eventuele wait op de opgaande flank van het CLK-signaal. Als wait weer hoog wordt, merkt de R800 dit ook alleen op bij een opgaande flank van de klok. Normaal is IORQ of MEMRQ maar een halve klokcyclus actief, maar kan met wait dus worden uitgerekt met hele klokcycli en niet met halve. De tijd tussen het actief worden van MEMRQ of IORQ en de eerstvolgende opgaande flank van het CLK-signaal is hooguit 50 ns op 40 MHz. De S1990 maakt na het actief worden van IORQ of MEMRQ — mits niet voor dram bestemd — het wait-signaal actief, maar wel met een vertraging van ongeveer 45 ns!
Die vertraging maakt op 40 MHz de timing van het wait-signaal zo kritisch, dat de propagatievertraging van een van de snelste OR-poorten (74AC32) al teveel is. Het te laat komen van het wait-signaal is dus de beperking van de klokfrequentie. Helaas heb ik nog geen da-taboeken van de AC-familie gevonden. Maar ik heb gehoord dat de propagatievertraging ongeveer gelijk is aan die van de 74AS-familie en die heeft een karakteriserende vertraging van l ns. In elk geval kan ik de vertraging van de poort niet meten met de scoop.
Mijn computer draait prima op de 33 MHz van Moonsound. Die heb ik toch ingebouwd, maar een frequentie van 38,97 MHz is te kritisch. Stel dat de OR-poort een vertraging heeft van 4 ns, dan zou die 50 ns voor het wait-signaal 54 ns worden:
40 MHz / 4 = 10 MHz
l /10 MHz = 100 ns
Een hele klokcyclus duurt dus 100 ns; een halve cyclus is die 50 ns. Te-rugrekenen naar de klokfrequentie: 54 ns, dus 108 ns komt overeen met 9,26 MHz, oftewel 37,037037 MHz.
In theorie is deze frequentie haalbaar, maar ik heb dit niet kunnen proberen, omdat ik geen oscillator of kristal heb voor deze frequentie.
Ik heb een kristaloscillator laten maken van 36,853333 MHz; dan kan ik de VCLK-uitgang van de R800 gebruiken om GFX9000 ook op te voeren. Dit zou betekenen dat de OR-poort een propagatievertraging van 4.26 ns mag hebben. Ook dit werkt prima. Maar de schakeling is ook heel goed te gebruiken op 28 MHz. Ook dan zijn de resultaten soms beter dan alleen een oscillator van 40 MHz. Bovendien zijn de problemen van het geluid en de VDP voorbij.



Figuur 3: nowait2

Refresh

Elke 24.3 s wordt de R800 2.6 s op non-actief actief gezet door de S1990, zodat de S1990 een refresh kan uitvoeren. Dit wordt in de MSX TurboR bepaald door de klokfrequentie van de VDP, dus wat dat betreft heb ik er zelf weinig voordeel van, omdat mijn VDP is opgevoerd. Hoe hoger die frequentie, des te vaker de R800 2.6 s lang van zijn werk wordt gehouden. Bij mij was dat elke 19.4 s. Maar is het nou werkelijk nodig zo vaak een refresh uit te voeren?
Ik moet als eerste opmerken dat ik andere dram's in mijn computer heb gezet. Deze zijn van 70 ns en misschien zo verbeterd, dat ze iets minder vaak hoeven te worden ge-refreshed. Het kloksignaal van de VDP gaat naar pin 17 van de S1990; dit lijntje heb ik doorgekrast en ik heb een frequentie van l MHz aangeboden aan de S1990. Houd er rekening mee dat de FM-PAC wel de 3.5 MHz van de VDP moet houden.



Figuur 4: refresh

Als dit werkt op de standaard-dram's is het leuk, maar ik denk niet dat het de moeite waard is de dram's ervoor te vervangen. Als je dit wel wilt doen, knip dan de pootjes eraf en haal de resten voorzichtig uit het moederbord. Ik heb dat niet gedaan, en toen

ik de eerste dram eruit trok, was ik even wanhopig. Hij bleef hangen aan 10 a 12 sporen richting R800.
De snelheidswinst is weliswaar niet erg groot, maar dat is de ingreep ook niet. De winst is zo'n 8%; dit betekent dat een MSX op 40 MHz de snelheid krijgt van 43 MHz. Met harddisksnelheid merk ik geen verschil, maar ik denk dat ik het maximum van mijn harddisk al heb bereikt. Met picview is het tekenen van een GIF-plaatje een paar seconde sneller, maar op een totale tijd van 35 seconde is dat niet enorm. Zelfs als de refresh eruit wordt gehaald, door gebruik van bijvoorbeeld cache ram in plaats van dram, zal de winst maar 10% zijn: van 40 MHz naar 44 MHz.

Conclusie

Het opvoeren met behulp van de schakeling nowait of nowait1 is vrij eenvoudig en geeft een redelijk resultaat. Het nadeel van het opvoeren van een cartridge, is dat er een draad van de het inwendige van de cartridge de computer in moet. Dit is eventueel op te lossen door een vrij pennetje van het slot te gebruiken, of in de computer zelf het ena-ble-signaal te genereren.
De methode van nowait2, direct van de R800-bus, is een iets grotere ingreep, maar het resultaat is zeer goed. Het probleem van deze methode is echter dat er voor cartridges minstens 12 draden van de cartridge richting R800 moeten. Je zou de cartridge kunnen inbouwen, maar het is handiger om de address- en control-bus die naar het slot gaan, door te krassen en de R800 er direct op aan te sluiten. Het enable-signaal kan dan in de computer zelf gegenereerd worden.
De refresh inkorten kan heel eenvoudig; die 8% winst is altijd iets. Dit inbouwen is maar weinig werk. Alleen de PCM en de counter van de S1990 lopen ook langzamer. De schema's nowait2 en refresh zijn de schema's die ik zelf in mijn computer heb gezet. Met een klokfrequentie van 36.9 MHz is het een behoorlijke vooruitgang

De hier beschreven schakelingen heb ik stuk voor stuk in mijn eigen MSX TurboR ingebouwd en het geheel werkt al enige maanden probleemloos. Toch ben ik niet aansprakelijk voor eventuele defecten die het gevolg kunnen zijn van het inbouwen ervan.