

Bennie Mols

Kijkduinstraat 121-2  
1055 XW Amsterdam  
bmols@wanadoo.nl

Studiegroep Wiskunde met de Industrie 2006

# Verdeel de software en verover de markt

**Apparaten die computerchips fabriceren, draaien op een kolossaal softwareprogramma. Drie tot vier maal per jaar verschijnt er een nieuwe versie. Is het niet handiger het programma in stukken te hakken? Elk stuk kan dan nog sneller als nieuwe versie op de markt verschijnen. De grote vraag is: wat is dan het optimale aantal stukken?**

Het Veldhovense bedrijf ASML is een van de grootste producenten ter wereld van lithografiemachines: machines die computerchips produceren. Deze machines schrijven met licht van verschillende (kleine) golflengten structuren op een siliciumschijf, de *wafer*. Hoe kleiner de golflengte van het licht, hoe kleiner de structuren. Eén enkele wafer bestaat uit een aaneengesloten patroon van dezelfde computerchips. Grote chipfabrikanten zoals Intel en Motorola maken gebruik van de asml-machines. De chipindustrie is zeer dynamisch. Al enkele decennia verdubbelt het aantal transistors op een computerchip elke twee jaar. De chipfabrikanten willen snel inspelen op nieuwe ontwikkelingen. Bij de lithografiemachines levert asml een softwarepakket voor de aansturing. Om te voldoen aan de steeds veranderende eisen van de markt (nieuwe machines en nieuwe functies), vernieuwt het bedrijf dat softwarepakket drie tot vier maal per jaar. De tijd tussen het begin van de ontwikkeling van een nieuw pakket en het moment dat het op de markt verschijnt, is ongeveer negen maanden. Het installeren van de nieuwe software op een lithografiemachine is kostbaar. Elk uur dat de machine stil staat, kost de chipfabrikant duizenden euro's. Om die reden willen niet alle afnemers van de

machines elke nieuwe versie installeren. Dat betekent voor asml dat er wereldwijd verschillende oudere versies circuleren van hun software. En het bedrijf wil die allemaal blijven ondersteunen. Bovendien hebben afnemers vaak helemaal geen behoefte aan een nieuwe versie van de complete software, maar zijn ze vaak alleen geïnteresseerd in een nieuwe versie van een deel van de software.

## Twintig miljoen regels code

“Ons huidige softwarearchief bestaat uit ruwweg twintig miljoen regels code”, vertelt Joost Smits van asml. In de loop van de ruim twintig jaar dat het bedrijf bestaat is die code telkens weer aangepast en verbeterd. Het bedrijf heeft momenteel vijfhonderd man in dienst alleen voor het softwaregedeelte van de machines. “Elke machine bestaat uit subsystemen die met elkaar moeten kunnen praten”, legt Smits uit. “Bijvoorbeeld de lens, de laser en de tafel met de wafer waarop de chipstructuren worden geschreven. Maar in de software zit nu alles aan alles vast. We zouden op dit moment niet eens in staat zijn om van elk subsysteem een nieuwe softwareversie op de markt te brengen. De komende twee jaar willen we daarom het softwarearchief in onafhankelijke stukken hakken.” Daarom wil asml

uitzoeken of het loont om het softwarearchief op te delen in modules, die het bedrijf ieder afzonderlijk in vernieuwde versie op de markt kan brengen. Het idee is dat een vernieuwde module sneller op de markt kan worden gebracht dan een vernieuwd compleet pakket, simpelweg omdat de module kleiner is. De voordelen van dit idee lijken echter groter dan ze zijn. De ontwikkelingstijd neemt inderdaad af met afnemende grootte van een module. Maar voor de testtijd van een module geldt iets gecompliceerd. Voor een deel van de tests die asml uitvoert, neemt de testtijd af met afnemende modulegrootte. Dat is logisch. Maar er zijn ook tests die een standaardlengte hebben, onafhankelijk van de modulegrootte. Voor dat testdeel neemt de tijd razendsnel toe als er meer modules komen. Omdat het bedrijf ook alle oude versies ondersteunt, moet elke nieuwe module ook getest worden in een omgeving van alle oude versies van alle andere modules. De ontwikkelingstijd plus de testtijd samen bepalen de marktintroductietijd: hoe snel na het ontwikkelingsbegin een module op de markt verschijnt. De opgave is nu om uit te zoeken wat het optimale aantal modules is, waarbij die marktintroductietijd minimaal is. Dat blijkt een gecompliceerd wiskundig probleem te zijn.

## Drie modules optimaal

“Het grote probleem is dat het optimale aantal modules zo sterk afhangt van welke aan-

names je doet”, vertelt wiskundige Jacques Resing van de TU Eindhoven. “Het hangt af van de grootte van de modules. Hebben alle modules gelijke grootte, of verschillen die? Ga je uit van gelijkblijvende mankracht om de modules te testen, of neem je meer mankracht in dienst. Ook hangt het af van hoe populair een bepaalde module is. Afhankelijk van de precieze aannames die we doen, vinden we andere oplossingen. Daarom zijn we voorzichtig met het trekken van harde conclusies.” Om het probleem te vereenvoudigen, namen de wiskundigen aan dat alle modules dezelfde grootte hebben, dat de beschikbare capaciteit gelijk blijft, en dat bekend is hoeveel vraag er naar een bepaalde module is. Zo namen ze als voorbeeld aan dat de populairste module twee maal zo populair is dan de op een-na-populairste, en drie maal zo populair als de op twee-na-populairste, enzovoort. “Dat is een soort wetmatigheid die je in de praktijk wel vaker vindt”, zegt Resing. “Bijvoorbeeld bij de populariteit van webpagina’s of bij het vóórkomen van veelgebruikte woorden in teksten.” Onder deze aannames is de voornaamste conclusie dat de marktintroductietijd minimaal is voor drie modules. “We hebben gerekend aan een paar voorbeelden”, legt Resing uit. “Dan zien we dat als je van 1 naar 2 naar 3 modules gaat, de marktintroductietijd bij elke stap afneemt. Maar hak je de software in meer dan drie stukken, dan begint de marktintroductietijd juist snel toe te nemen. Dat komt omdat je elke nieuwe module steeds moet testen tegen alle oude versies van alle andere modules. Dan weegt de kortere ontwikkeltijd voor een kleinere module niet langer op tegen de steeds langer wordende testtijd.” Wat betekent het in drie stukken hakken voor het aantal nieuwe versies dat dan per jaar op de markt kan verschijnen? Resing: “Een populaire module kan zo’n vijf tot zes keer per jaar uitkomen. De klant gaat er dus op vooruit, omdat hij sneller nieuwe eigenschappen in de software heeft dan in de oude situatie. Voor een minder populaire module is een of twee maal per jaar genoeg.” Het optimale aantal modules blijkt sterk af te hangen van hoe populair diverse modules zijn. Het kan ook zijn dat er een groot verschil is tussen hoeveel vraag er is naar de populairste module en naar de minst populaire module. En wel een veel groter verschil dan in de wetmatigheid die ook voor de populariteit van webpagina’s blijkt te gelden. In dat geval kan het optimale aantal modules zelfs toenemen tot vijf. De minst populaire module hoeft dan maar eens in de drie jaar te worden vernieuwd, de op-een-na minst populaire eens

in de twee jaar, de volgende eens per jaar, en ten slotte kunnen de twee meest populaire modules maar liefst elke maand worden vernieuwd. “Wij bevelen asml aan om eerst heel goed te inventariseren hoe sterk de vraag is naar de verschillende modules, voordat ze het optimale aantal modules bepalen”, aldus Resing.

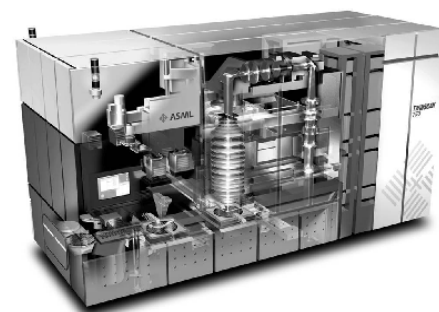
### Paarsgewijs testen

Omdat de testtijd zo snel blijkt toe te nemen bij meer modules, hebben de wiskundigen nagedacht over een slimme manier om die testtijd te beperken. Een handige truc blijkt het zogeheten paarsgewijs testen. Het is bekend dat de meeste softwareconflicten ontstaan door twee modules die na vernieuwing niet langer meer goed met elkaar kunnen samenwerken. Veel minder vaak komt het voor dat er softwareproblemen ontstaan doordat drie modules samen niet meer goed blijken te kunnen functioneren. Met die informatie valt het aantal tests flink terug te brengen. Resing: “Dat betekent dat je niet langer elke nieuwe module tegen alle versies van alle andere modules hoeft te testen, maar dat je alleen nog maar naar paren van modules hoeft te kijken. Bijvoorbeeld alleen module 1 en 2, module 2 en 3, module 3 en 4, enzovoort. De test van bijvoorbeeld 1,2,3 of van 2,3,4, enzovoort, kun je allemaal weglaten. De kans dat het daar misgaat is immers klein in de praktijk.” Stel dat drie modules elk drie versies hebben die in de afgelopen tijd op de markt zijn gekomen. Het testen van elke versie van een module van elke andere versie van elke andere module, levert dan 27 tests. Bij paarsgewijs testen blijven er maar negen test over. Bij vijf modules met elk vier versies brengt paarsgewijs testen het aantal tests zelfs terug van 1024 naar 23. De winst van slim testen is dus groot.

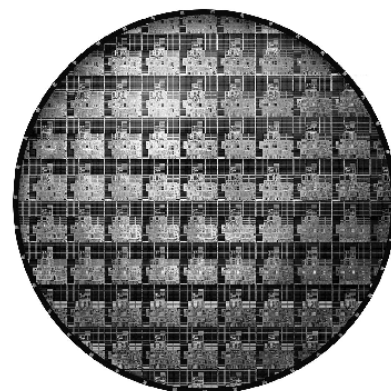
“Voor ons was het heel goed dat een groep wiskundigen met een heel andere blik tegen ons probleem aankijkt”, vertelt Joost Smits van asml. Zuiver en alleen door op technische gronden naar de software te kijken, had asml het vermoeden dat ze het hele pakket in drie stukken konden hakken, die elk afzonderlijk als module op de markt zouden kunnen verschijnen. Smits: “Dat is hetzelfde aantal als het resultaat van de studiegroep. Maar wij hadden daar totaal geen wiskundige onderbouwing voor. Het is mooi om te zien dat twee verschillende manieren van kijken hetzelfde resultaat opleveren. Op basis van de wiskundige resultaten weten we nu ook dat het belangrijk is om meer gegevens te verzamelen over hoe vaak onze klanten een nieuwe versie willen hebben van een bepaalde module.

Het eerste wat we gaan doen is om onze code, waarin alles met alles samenhangt, uiteen te rafelen in onafhankelijke stukken. Daarna kunnen we dan bekijken hoeveel modules we echt op de markt willen brengen.”

**Meer informatie**  
www.asml.com



**Figuur 1** Een asml Twinscan lithografiemachine. Duidelijk zichtbaar is het lichtpad door de enorme lens met al zijn elementen. Onderin zijn de twee waferposities te zien (vandaar de term Twin in de naam). In de eerste positie worden de wafers opgemeten en in de tweede worden ze belicht. (foto: asml)



**Figuur 2** Een siliciumwafer met computerchips. Een asml-lithografiemachine brengt met licht de structuren aan op zo’n siliciumwafer. (foto: www.intel.com)

Microprocessor	Year of Introduction	Transistors
4004	1971	2,300
8008	1972	2,500
8080	1974	4,500
8086	1978	29.000
Intel286	1982	134.000
Intel386™ processcr	1985	275.000
Intel486™ processor	1989	1.200.000
Intel® Pentium® processor	1993	3.100.000
Intel® Pentium® II processor	1997	7.500.000
Intel® Pentium® III processor	1999	9.500.000
Intel® Pentium® 4 processor	2000	42.000.000
Intel® Itanium® processor	2001	25.000.000
Intel® Itanium® 2 processor	2003	220.000.000
Intel® Itanium® 2 processor (9MB cache)	2004	592.000.000

**Figuur 3** Het exponentieel toegenomen aantal transistors op een computerchip in de loop van de tijd (bron: www.intel.com)