Bilgisayar Mimarisinde Yeni Yaklaşımlar

Tekrar Düzenlenebilir Mimariler (Reconfigurable Architectures)

Selçuk Giray Özdamar
504061529

İÇİNDEKİLER

GİRİŞ

Bilgisayar mimarileri, bilgisayar bilimlerinde ve akademik çevrelerde en çok üzerinde durulan ve çalışılan araştırma konularından birisi olmuştur. Araştırmalar; ortaya çıkan ekipmanın fiyatı, sistemin programlanabilirliği, sistem yayımlandıktan sonra (deployment) üzerinde çalıştığı ortamın özellikleri ve daha birçok farklı amaçla yapılmaktadır.

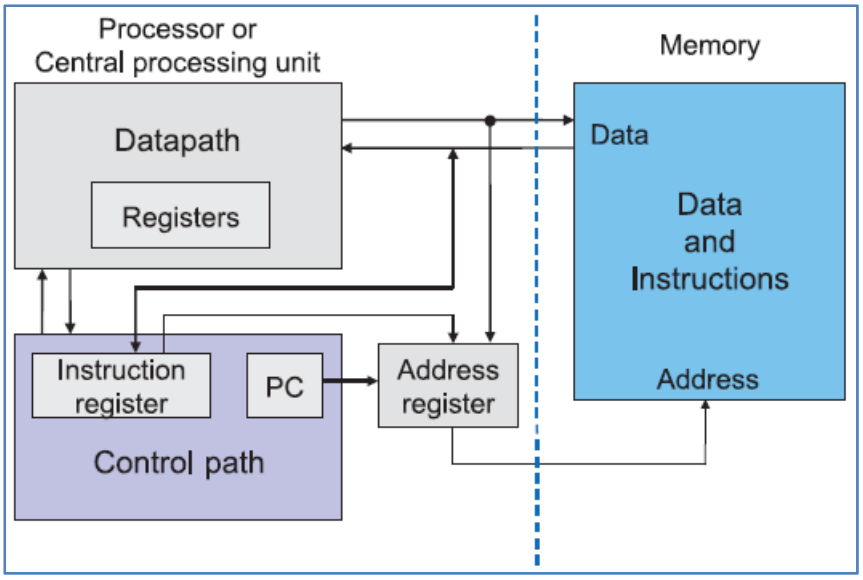
Yüksek başarımlı paralel sistemlerde kullanılan işlemcilerde (hava durumu simülasyonu vb.), yüksek saat oranı (high clock rates), paralellik ve hızlı bir iletişim bandgenişliğine odaklanılmıştır. Gömülü cihazlar, küçük bir mikrodenetleyici (microcontroller) sensörlerden alınan verilerin kontrolünü sağlarken, aynı zamanda düşük frekanslarla bu verileri işleticilere aktarır. Gömülü cihazlarda temel amaç güç sarfiyatı optimizasyonu ve fiyatın eniyilemesini (optimizasyonunu) sağlamaktır.

Bulunduğu ortamı keşfetmesi ve incelemesi amaçlanan otonom bir kart (autonomous cart) için, işlemci birimi görüntüleri yakalayıp, uygun şekilde sıkıştırdıktan sonra bunları temel bir kontrol noktasına gönderir. Buna paralel olarak, sistem aynı zamanda engel tanıma ve kaçınma (obstacle detection and avoidance) yeteneğine sahip olmalıdır. Bu şekildeki bir sistemin güç tüketimi sistemin uzun süre ayakta kalabileceği şekilde optimize edilmelidir. Diğer taraftan önemli görüntülerin kaçırılmaması için, işlemci görüntüleri olabildiğince hızlı işlemelidir. Engel tanıma ve kaçınma hızlı bir şekilde yerine getirilmelidir. Amaçların çokluğu farklı işlemci mimarileri tasarlamayı gerektirmiştir, her bir mimari yaklaşım amaca yönelik olarak optimize edilmiştir. Esneklik (flexibility) bakımından mimariler üç ana grup altında sınıflandırılabilir: Von Neumann (VN) örneğini temel alan genel amaçlı mimariler, büyük oranda aynı karakteristiklere sahip uygulamalar için tasarlanmış domain özel (domain-specific) işlemciler, tek bir uygulama için tasarlanmış uygulama-özel (application-specific) işlemci mimarileri.

Genel Amaçlı Mimari

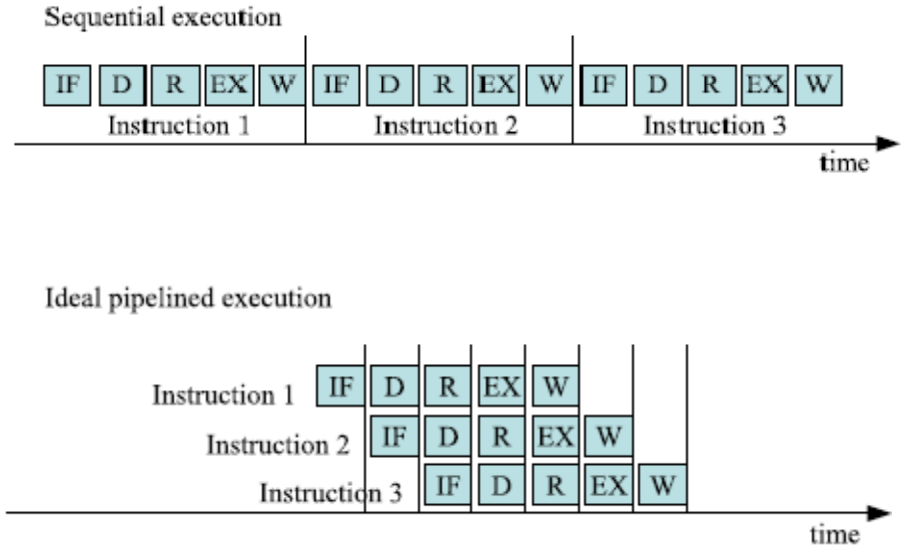
1945 yılında, John Von Neumann adlı matematikçi, basit, sabit yapılı ve donanıma değişiklik yapmadan verilen herhangi bir hesaplamayı yerine yapabilecek bir bilgisayar tasarlamıştır. VN yaklaşımı evrensel olarak kabul görmüş, gelecek nesil yüksek-hızlı sayısal bilgisayarlar esas alınmış ve uygulanmıştır. VN yönteminin kabul görmesindeki temel neden, basit olması ve kolay programlanabilir olmasıdır. Şekil 1.1'de VN makinesinin genel yapısında görüldüğü üzere:

- Program ve verileri saklayan bellek (memory). Harvard mimarisinde paralel erişilebilen iki bellek bulunur, bunlardan birisi programı diğeri veriyi muhafaza eder.
- Denetim birimi (control unit, control path), işletilecek bir sonraki komutun adresini saklayan program sayacından (program counter) oluşur.
- Aritmetik ve mantıksal birim (arithmetic logic unit, ALU) komutların işletildiği birimdir.



Şekil 1.1 Von Neumann Bilgisayar Mimarisi

VN bilgisayarında genel olarak bir komutun işletimi toplam 5 adımda tamamlanır: Komut okuma (Instruction Read, IR) adımında komut bellekten okunur; Komut çözme (Decoding, D) adımında komutun ne anlama geldiği çözülür ve operandların yeri belirlenir; Operand okuma (Read Operands, R) adımında operandlar bellekten okunur; Yürütme (Execute, EX) adımında komut okunan operandlar ile birlikte işletilir; Sonuç yazma (Write Result, W) adımında yürütmenin sonucu belleğe geri yazılır.



Şekil 2 : Von Neumann bilgisayarında komutların sıralı ve pipeline işletimi

VN bilgisayar mimarisinin temel avantajı esnek olmasıdır, nedeni ise neredeyse bütün algoritmalar için programlanabilir olmasıdır. Ancak, her algoritma VN kurallarına göre kodlandığı takdirde VN bilgisayarı üzerinde çalıştırılabilir. Burada söylenmek istenen , "algoritma kendini donanıma göre uyarlamalıdır". Farklı uygulamalar tarafından aynı

donanımın geçici (temporal) olarak kullanılmasından dolayı VN yöntemi, geçici hesaplama (temporal computation) olarak karakterize edilmektedir.

Çalıştırılan algoritmaların türü önceden biliniyor olsaydı, bu algoritma türünün hesaplama yöntemine göre işlemci en iyi şekilde uyarlanabilirdi. Bu durumda, veri yolunun (data path) bellekten komut yakalama ve komut çözme işlemleri gereksiz olacaktı. Diğer taraftan, verinin kaynağı ve yeri önceden bilindiği durumda bellekten okuma ve belleğe geri yazma işlemleri de gereksiz olacaktı.

Domain-Özel İşlemciler

Domain özel (domain-specific) işlemciler aynı karakteristiklere sahip algoritmalar için uygun hale getirilmiş işlemcilerdir. Bir önceki bölümde anlatıldığı gibi, veri yolu (data path) benzer türdeki algoritmaların yaygın kullanılan operasyonlarının ideal işletimi için uygun hale getirilmiştir.

Sayısal sinyal işlemcisi (DSP) tekrarlayan ve sayıca yoğun hesaplamaların hızlandırılması amacıyla sinyal işleme alanında özellikle telekominikasyon, multimedia, otomobil, radar, sonar, sismik, görüntü işleme alanlarında oldukça yaygın olarak kullanılan özelleştirilmiş bir işlemcidir. DSP'lerin en çok bahsedilen özelliği, bir (veya birden fazla) multiply accumulate (MAC) operasyonunu tek bir çevrimde tamamlayabiliyor olmasıdır. Genelde MAC işlemleri büyük veri setleri üzerinde uygulanır. Bir MAC işleminde veri ilk önce çarpılır, daha sonra birikmiş (accumulated) veri üzerinde toplama yapılır. Normal VN bilgisayarında MAC işlemi toplam 10 adımda tamamlanır. İlk komut (multiply) yakalanır, çözülür, operandları bellekten okunur, çarpılır ve sonuç belleğe yazılır; sonraki komut (accumulate) okunur, bir önceki komutun sonucu bellekten okunur, toplanmış değerin üzerine eklenir, belleğe geri yazma işlemi yapılır. Sayısal sinyal işlemcisi (DSP) kendi özel donanımı sayesinde bu adımların hepsini doğrudan (belleğe erişmeksizin) gerçekleştirir.

Sayısal sinyal işlemcileri (DSP) veri genişlikleri bakımından farklı uygulama etki alanları için uyarlanabilir özelliğe sahiptir. Örneğin, görüntü işleme amaçlı bir DSP'de, pixel'ler üzerinden işlem yapmak gerekmektedir. Pixel'in RGB (kırmızı yeşil mavi) olarak üç renkle temsil edildiğini düşünürsek, ve her bir rengin 1 byte ile temsil edildiği düşünüldüğünde, görüntü işleme amaçlı kullanılan bir DSP'de veri yolunun (data path) 8 bit'den daha büyük olması anlamsızdır. Açıkçası, görüntü işleme amaçlı kullanılan bir DSP bu durumda 32 bit hesaplama gerektiren uygulamalar için kullanılamaz hale gelecektir.

DPS'nin bu şekilde özelleşmiş olması başarımı artırmasına ve kullanımı geliştirmesine karşın, farklı amaçlı uygulamalar için kullanılamaması nedeniyle esneklik (flexibility) indirgenmiş olmaktadır.

Uygulama-Özgü İşlemciler

Sayısal sinyal işlemciler (DSP) belli bir dereceye kadar uygulama- özgü (application-specific) özellikler içermesine rağmen (örneğin: multiply accumulate işlemi, veri genişliği optimizasyonu) , yine de VN yöntemini kapsamaktadır, o yüzden hala sıralı (sequential machine) makine olarak kalmaktadır, performansları sınırlıdır. Bir işlemci önceden belirlenmiş ve sabit bir uygulama için tasarlandığında ve sadece bu uygulama için kullanılıyorsa, işlemci birimi bu uygulama için optimize edilebilir. Bu "donanımı uygulamaya adapte etme" şeklinde söylenebilir. Çoklu ortam işlemede (multimedia processing), işlemciler görüntü sıkıştırma amaçlı olarak görüntü sıkıştırma standardına uygun olarak tasarlanırlar. Bu tür işlemciler görüntü sıkıştırma dışında başka bir amaç için kullanılamaz. Sadece belli bir uygulama için tasarlanan işlemcilere "uygulama-özel işlemciler" (Application-Specific Processor, ASIP) adı verilmektedir. ASIP'de komut çevrimleri (IR, D, EX, W) ortadan kaldırılmıştır. Komut seti doğrudan donanımın üzerinde gerçeklenmiştir. Veri girişlerinden gelen veri işlemcide gerekli şekilde işlendikten sonra, sonuçlar işlemcinin çıkışından alınabilir. Uygulama-özel işlemciler genelde tek bir chip olarak tasarlanırlar. Buna "uygulama-özel entegre devre" (Application-Specific Integrated Circuit, ASIC) adı verilir.

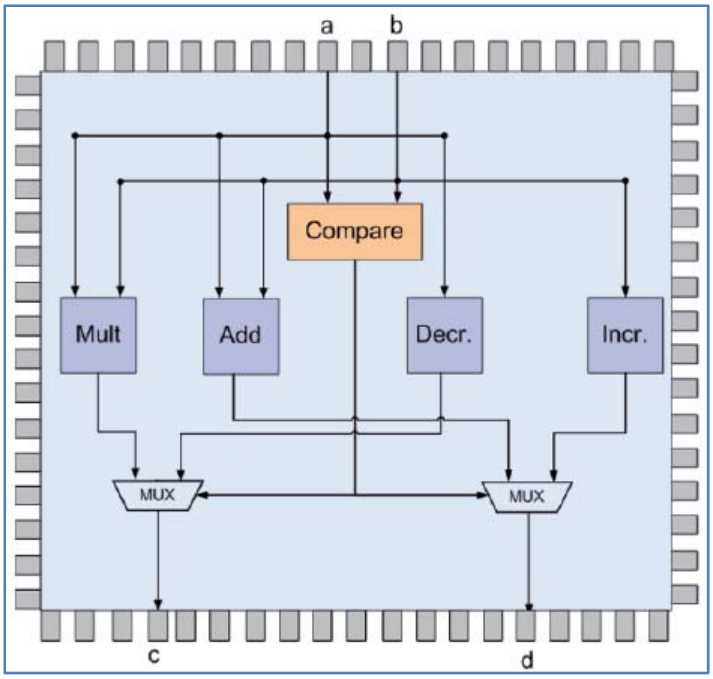
Örnek 1.1: *Algoritma 1'in Von Neumann bilgisayarında çalışması isteniyorsa, en az 3 komuta ihtiyaç vardır.*

Algoritma 1

```
if a < b then
      d = a + b
      c = a x b
else
      d = b + 1
      c = a - 1
end if
```

Komut çevrim süresi t_{cycle} olarak kabul edersek, program $3 * 5 * t_{cycle} = 15 * t_{cycle}$ sürede işletilecektir (pipelining olmadığı varsayılıyor).

Aynı algoritmanın ASIP'de gerçeklendiğini hesaba kattığımızda; $d = a + b$ ve $c = a * b$ işlemlerinin paralel olarak gerçeklenebilir olduğu görülüyor. Aynı durum d = b + 1, c = a – 1 için de geçerlidir. Şekil 3'de ASIP gerçekleştirimi gösterilmektedir.



Şekil 3: Algorithm 1'in ASIP gerçekleştirimi

İlk aşamada a+b, a∗b, b+1, a−1 komutları, hatta a < b karşılaştırma işlemi bile paralelde yürütülebilir. a < b karşılaştırmasının sonucuna göre, ilk aşamada elde edilen sonuçlar programda gösterildiği şekilde c ve d'ye aktarılır. İşlemcide bir sinyalin en uzak iki nokta arasında bir noktadan diğerine fiziksel iletimi için gereken maksimum sürenin t_{max} olduğunu kabul edelim (input-multiply-multiplex yolu üzerinde bu durum gerçekleşir). t_{max} aynı zamanda ASIP işlemcinin çevrim süresidir. İki adet a ve b girişleri için sonucun c ve d'den alınması t_{max} sonunda gerçekleşir. VN işlemcisinde sonucun c ve d'den alınabilmesi için gerekli süre 15 * t_{cycle} olduğuna göre, VN işlemcisinin ASIP ile boy ölçüşmesi için

15 * $t_{cycle} < t_{max}$, $t_{cycle} < t_{max}$ / 15 eşitliğinin sağlanması gerekir. VN işlemcisinin ASIP işlemcisinden en az 15 kat daha hızlı olması gerekir. Bu örnekte VN makinesinde pipeline olmadığı varsayılmıştır. Pipeline olduğu durumda da karşılaştırma benzer şekilde ele alınabilir.

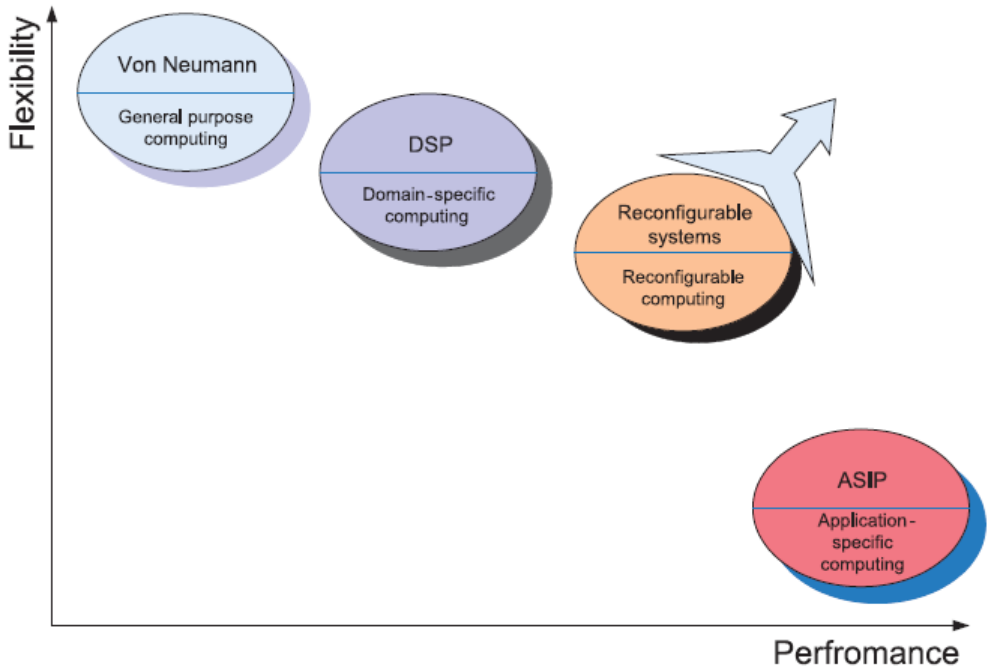
Uygulama-özel işlemciler (ASIP) belirli bir uygulamayı gerçeklemek için sınırlı-bölgesel yaklaşımı (spatial approach) kullanır.Uygulamanın tüm parçalarının hesaplanabilmesi için gerekli fonksiyonel birimlerin işlemcinin üzerinde bulunması gerekir. Bu tür hesaplama yöntemine sınırlı-bölgesel hesaplama (Spatial Computing) adı verilir. Tekrar hatırlatmak gerekirse, belirli bir uygulama için tasarlanan uygulama-özel işlemciler (ASIP) başka görevler için kullanılamaz.

Tekrar-Düzenlenebilir Mimari

Önceki bölümlerde üç farklı işlemci yaklaşımı anlatıldı. İşlemcileri karakterize etmek için esneklik (flexibility) ve başarım (performance) şeklinde iki temel ölçüt olduğunu anlamış olduk.

- The VN mimarisi üzerinde birçok farklı uygulama çalıştırabiliyor olmasından dolayı çok esnek yapıya sahip. Bu sebepten dolayı VN mimarisine "genel amaçlı işlemci mimarisi" (General Purpose Processor, GPP) adı verilmektedir. Performans açısından çok da iyi olduğu söylenemez çünkü paralel hesaplama yapamıyor. Bundan başka, bir komutun işletimi için gerekli beş adımdan oluşan çevrim (IR, D, R, EX, W) büyük bir dezavantaj teşkil etmekte, büyük veri kümeleri için bir komutun işletimi de önemli bir sorun oluşturmaktadır. İşletim yapmak için gerekli kıstas "algoritma kendini donanıma göre uyarlamalıdır" şeklinde olduğu, esnektir.
- Uygulama-özel işlemciler (ASIP) daha yüksek başarıma sahiptir, çünkü belirli bir uygulama için optimize edilmiş mimarilerdir. Uygulama için gerekli komut seti yonganın üzerine yerleşiktir. "donanımı her zaman uygulamaya adapte et" yaklaşımı ile başarım elde edilmektedir.

İki ölçüt de göz önüne alındığında, bir tarafta başarım (performance) diğer tarafta esneklik (flexibility) bulunmaktadır. Şekil 4'de görüldüğü gibi VN mimarisi bir tarafta, ASIP mimarisi ise diğer tarafta yer almaktadır.

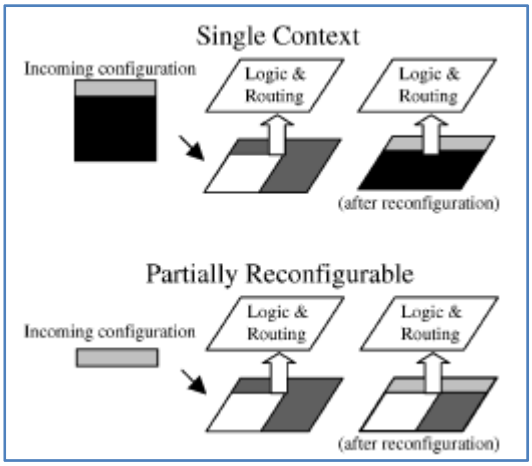


Şekil 4 : Esneklik (Flexibility) vs Başarım (performance)

Hesaplama kriterlerine göre uygun bir işlemci nasıl seçilmelidir? Uygulamanın sınırları tam belli değilse,veya birçok uygulama için kullanılacaksa genel amaçlı işlemci mimarisi seçmek en mantıklı çözümdür. Ancak, gömülü bir cihazda çalışacak bir uygulama için gerekli bir işlemci için en iyi çözüm bu uygulama için optimize edilmiş yeni bir işlemci (ASIP) tasarlamaktır.

İdeal olarak, genel amaçlı mimarilerdeki esneklik ve uygulama-özel mimarilerdeki başarımın her ikisini de elde etmek mükemmel bir çözüm olacaktır. Uygulamaya anında adapte olabilen bir aygıt iyi bir çözüm olabilir. Bu şekildeki donanıma tekrar-düzenlenebilir donanım (reconfigurable hardware, reconfigurable device veya reconfigurable processing unit RPU) adı verilmektedir.

Belirli bir anda belirli bir uygulama için, en iyi hesaplama yöntemini uygulayarak hesaplama hızını artırmak amacıyla, makinenin sınırlı-bölgesel yapısı (spatial structure) değişikliğe uğrar. Aynı donanım üzerinde bu sefer farklı bir uygulama çalıştırıldığında, yeni uygulama ile eşlenebilmesi için donanım tekrar düzenlenmelidir. VN bilgisayarı ile karşılaştırıldığında, VN komut seti kullanılarak oluşturulan programı işletirken; tekrar-düzenlenebilir mimarilerde donanımın yapısı (tamamı veya bir bölümü) derleme zamanında (compile-time) veya çalışma zamanında (run-time) değiştirilebiliyor. Tekrar-düzenlenebilir mimarilerde donanım yapısının değiştirilmesi, donanıma ikil akıntı (bitstream) yükleme ile sağlanır.



Şekil 5 : Tam (fully) & Parçalı (partial) tekrar-düzenlenebilir yaklaşım

Donanımı tekrar düzenleme işlemi, çalışma başlangıcında (start-up-time) ve çalışma zamanında (run-time) yapılır.

Uygulama Alanları

- Rapid Prototyping
- In-System Customization
- Multi-modal Computation
- Adaptive Computing Systems

TEKRAR-DÜZENLENEBİLİR MİMARİLER

Bir önceki bölümde, tekrar-düzenlenebilir donanıma sahip bir cihazın (reconfigurable device) kendini algoritmaya dinamik olarak uydurabilmesinden bahsettik. Fabrikasyon aşamasında sabit şekilde tasarlanan donanım, yaşam süresi boyunca değişmezken, nasıl oluyor da uygulamanın gereksinimlerine göre çalışma zamanında dinamik olarak tekrar uyarlanabiliyor? Bu bölümde tekrar-düzenlenebilir donanımlarda çoğunlukla kullanılan mimarilere değinerek bu konuya ışık tutacağız.

Tekrar-düzenlenebilir mimarilerin (reconfigurable architectures) gelişiminden kısaca bahsettikten sonra birinci kısımda "ince taneli tekrar-düzenlenebilir donanım" lardan (fine-grained reconfigurable hardware) bahsedilecektir. Bu tür donanımların işlevselliği çok düşük seviyede taneli yapı düzeyinde (low level of granularity) değiştirilebilmektedir. Örneğin, donanıma yeni bir çevirici (inverter), veya iki girişli bir NAND-kapısı ekleyerek (veya çıkararak) donanımda değişiklik gerçekleştirilmiş olur. İnce taneli tekrar-düzenlenebilir donanımlar (fine-grained reconfigurable hardware) temelde "programlanabilir mantıksal donanım" (programmable logic devices, PLD) olarak temsil edilirler. Bunun içerisine PLA (programmable logic arrays), PAL (programmable array logics), CPLD (complex programmable logic devices) ve FPGA'ler (field programmable gate arrays) girer.

İkinci kısımda iri-taneli tekrar-düzenlenebilir donanımlar (coarse-grained reconfigurable hardware) bahsedilecektir. Bu tür donanımlar genellikle iri-taneli elemanlardan oluşurlar, veri akış işlevini (dataflow function) etkin şekilde sağlayan ALU örnek olarak verilebilir.

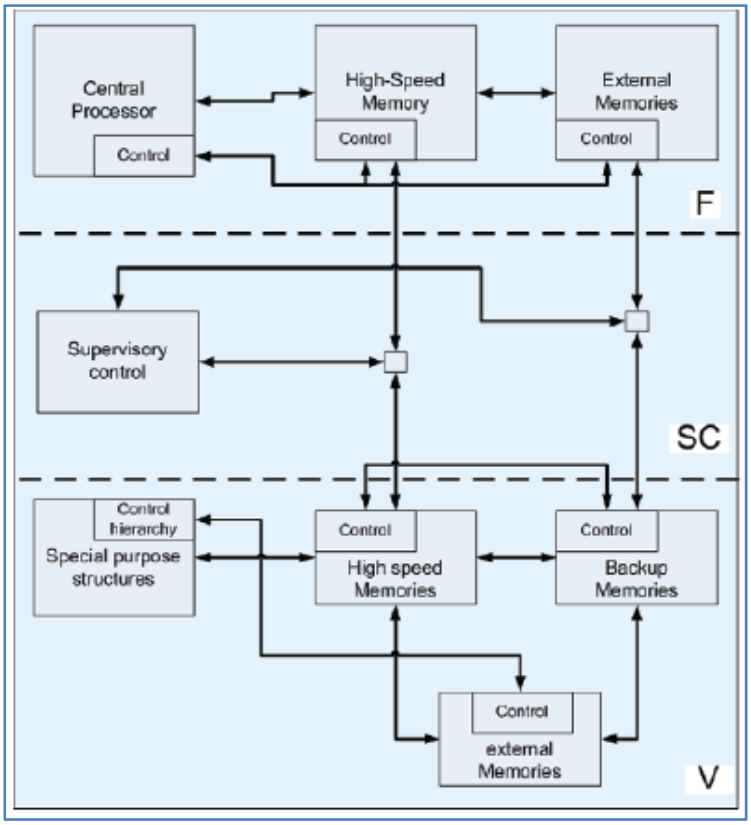
İlk Çalışmalar

Çalışma zamanında dinamik değişebilen donanım yapısına sahip mimari yaklaşım, diğer mimari yaklaşımlar gibi çok eskiye dayanmaktadır. İlk bilgisayar ENIAC'ın esnek olmayan yapısının üstesinden gelmek için Jon Von Neumann üç bloktan oluşan (memory, datapath ve control path) ve iyi kodlanmış bir programı işletebilen evrensel bir bilgisayar mimarisi önermiştir. Jon Von Neumann bu mimariyi sunmadan önce, çalışma zamanını daha çok herhangi bir uygulama için en iyi hesaplama yapısının nasıl olması gerektiği konusuna harcamıştır.

The Estrin Fix-plus Machine

1959 yılında California Üniversitesi'nde görevli bilgisayar bilimcisi Gerald Estrin tarafından tekrar-düzenlenebilir hesaplama (reconfigurable computing) kavramı ilk kez ortaya atılmıştır. 1960'da Estrin fix-plus machine adı verilen makineyi geliştirmiştir. Estrin'in tekrar-düzenlenebilirlik paradigması "değişken yapıdaki bilgisayara farklı uygulamalar için farklı konfigürasyon uygulanarak hesaplama hızlarından kazanç elde etmek mümkündür. Donanıma bu yetenek ancak programlanarak veya fiziksel olarak tekrar yapılandırarak verilebilir." üzerinde durmaktadır. Estrin vizyonunu gerçekleştirmek için fix-plus makinesini gerçekledi. Günümüzdeki birçok tekrar-düzenlenebilir makineler gibi, fix-plus makinesi de temel üç elemandan oluşmaktadır:

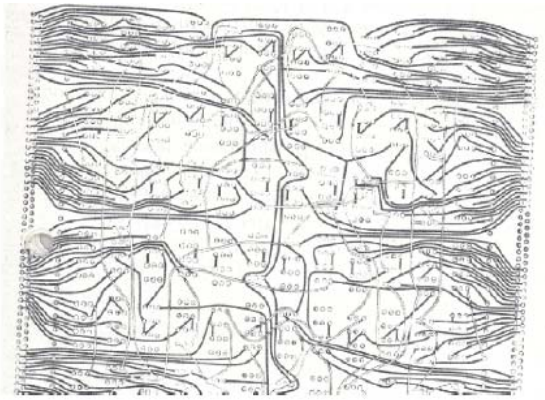
- Yüksek hızdaki genel amaçlı bir bilgisayar, sabit kısım (fixed part, F). Bu kısım genel amaçlı bir işlemci ile gerçeklenmiştir, IBM 7090 kullanılmıştır.
- Değişken kısım (variable part, V) ; farklı problemler için özel amaçlı konfigürasyonlar elde edebilmeyi sağlayan, farklı boyuttaki tekrar düzenlenebilir sayısal altyapılardan oluşur.
- Denetimsel kısım (supervisory control, SC) sabit kısım ile değişken kısmın koordinasyonunu sağlar.



Şekil 6 : Estrin Fix-Plus Machine Genel Yapısı

Değişken kısım (variable part, V) temel konfigürasyonunda optimize edilmiş fonksiyonel birimlerden (trigonometric functions, logarithm, exponentials, n-th power, roots, complex arithmetic, hyperbolic, matrix operation) oluşur.

Temel öbek modülleri (basic block modules) anakart üzerinde 36 farklı noktaya iliştirilerek farklı belirli bir uygulama için işlevsellik sağlanmış olur. Anakartın işlevselliği, temel öbek modüllerin yerine manuel olarak yenileri takılarak sağlanmaktadır.



Şekil 7: Estrin-Machine kablo demeti (wiring harness)

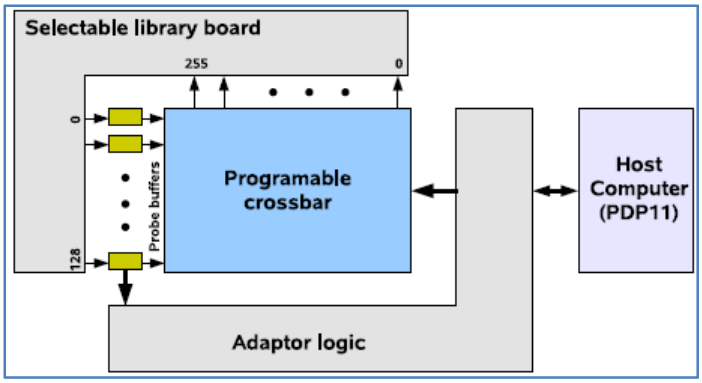
Modüller arası bağlantı yukarıdaki şekilde görüldüğü gibi kablo demeti ile sağlanmaktadır. Anakarttaki modüllerin yerine yeni modüller takılarak veya kablo demeti değiştirilmesiyle modüller arası yeni bağlantılar elde edilmesiyle tekrar-düzenleme sağlanır.

The Ramming Machine

1977 yılında Dortmund Üniversitesi'nde görevli araştırmacı Franz J. Rammig, donanımın düzenlenmesi fikrini önerdi. Amaç, mekanik veya manuel müdaheleler araya girmeden, gerçek sayısal bir donanımın şekillendirilmesidir.

Ramming fikrini gerçekleştirmek için bugünkü FPGA mimarilerindeki gibi bir donanım editörü tasarladı. Editör, modüllerden, bacaklardan ve bacaklara bire bir eşlenebilen fonksiyonlardan oluşmaktadır. Belirli bir fonksiyonun devre sistemi "string" olarak tanımlanabilmektedir. String ifadede iki harfli alfabede (w="wired" ve u = "unwired") bulunan karakterler yer alır. Bir donanım editörü (hardware editor) oluşturmak için, modüllerin çıktısı (output), seçicilere girdi (input) olarak verilir, seçicilerin çıktısı (output) ise modüllere girdi (input) olacak şekilde bağlantı kurulur. Ayrıntılı sistem mimarisi aşağıdaki şekil 9'da gösterilmiştir.

{wired, unwired} gerçekleştirimi, seçici dizisinden oluşan programlanabilir sürgü anahtarı (programmable crossbar switch) aracılığıyla yapılır. Bit dizgisi (Bit string) , seçiciyi(selector) kayıtçıda (register) depolanması ile kayıtçıdan sağlanır, ve aynı zamanda anasistemden (host computer) kayıtçılara erişebilmektedir. Modüller (Estrin'in Fix-Plus makinesinde olduğu gibi) kütüphane kart (library board) üzerinde bulunmaktadır. İstenilen kütüphane kartı yazılım kontrolüyle seçilebilir. Modülün giriş/çıkışları (I/O) ile bacaklar arasındaki eşleme manuel olarak yapılmıştır, ancak ilgili kart kütüphaneden yazılım vasıtasıyla seçilerek , uygun kart seçilerek tekrar-düzenleme yapılmış olur. Kütüphanedeki her bir kartın kablolaması ve bağlantıları sabittir, ancak ilgili kart seçildiği takdirde tekrar-düzenleme gerçekleşir.



Şekil 8 : Rammig machine genel yapısı

Rammig makinesi öykünüm altyapılarında (emulation platform) yoğun olarak kullanılmaktadır. Bugünkü FPGA'lerin de en yaygın kullanım alanı öykünüm altyapılarıdır.

Ayrıca anlatılan bu iki çalışmadan başka aşağıdaki çalışmalar da yapılmıştır:

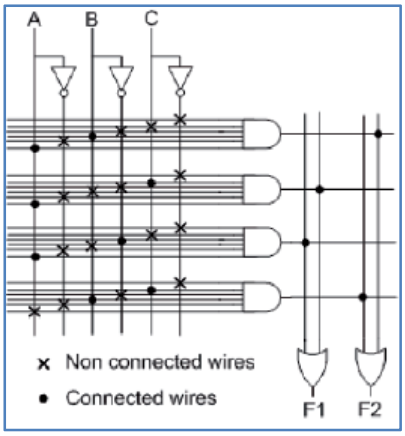
- Hartenstein's Xputer
- The PAM Machine
- The SPLASH II
- The Garp Approach

Basit Programlanabilir Mantıksal Devre

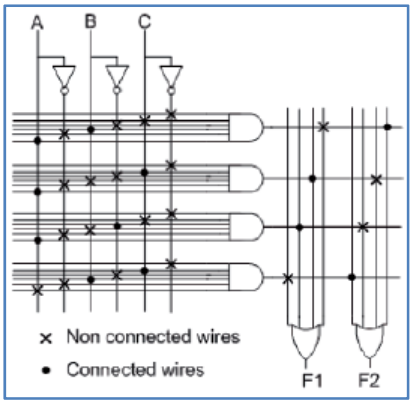
Programlanabilir mantıksal diziler (Programmable logic arrays, PLA) ve programlanabilir dizi mantıkları (programmable array logic, PAL) birbirine bağlı AND-kapıları ve OR-kapılarından oluşur. Giriş sinyalleri (aynı zamanda olumsuzlamalar "negation") AND-düzleminde bulunan AND-kapılarına girdi teşkil eder. AND-kapılarının çıktısı OR-düzlemindeki OR-kapılarına girdi olarak gelir. AND-düzlemi ve OR-düzlemi arasındaki bağlantılar kullanıcı tarafından programlanabilir.

Boolean bir fonksiyonun çarpımlar toplamı (sum of products) olarak ifade edilebilmesi sayesinde, PLA ve PAL'ler boolean bir fonksiyonun programlanması için kullanılabilir. Çarpımlar AND-düzleminde oluşturulan bağlantılarla, çarpımların toplamları ise OR-düzleminde bağlantılar oluşturulması ile sağlanır. PLA'larda her iki kapı düzlem de (AND ve OR düzlemi) programlanabilmesine karşın, PAL'de sadece AND düzlemi programlanabilir. OR düzlemi, üretici tarafından sabitlenmiştir. Bu nedenle PAL'ler PLA'nın alt sınıfı olarak görülebilir.

Örnek 2.1 Şekil 10 ve 11'de F1 = A * C + A * ^B and F2 = A * B + B * C fonksiyonlarının PLA ve PAL gerçekleştirimleri gösterilmektedir. PAL'de OR-düzlemi programlanamazken, PLA'da OR-düzlemindeki bağlantılar değiştirilerek farklı çarpımlar toplamı elde edilebilmektedir.



Şekil 9 : PAL, Sabitlenmiş OR-bağlantıları



Şekil 10 : Programlanabilir OR-bağlantıları PLA

Geliştirmeye açık şekilde, OR çıktısı flip-flop'a veya AND-kapısına girdi olarak beslenebilir. Bu sayede daha karmaşık devreler elde etmek mümkündür. PLA ve PAL iki seviyeli devre gerçekleştirmek için uygun donanımlardır. Örneğin, birinci seviyede çarpımlar elde edilirken, ikinci seviyede çarpımların toplamı elde edilir.

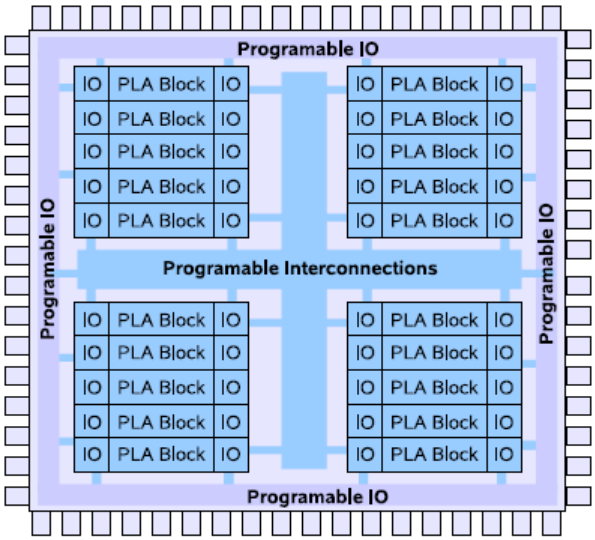
PLA ve PAL'lerde ana kısıt AND ve OR düzlemleri nedeniyle düşük kapasiteli olmalarıdır. Giriş sayısı arttıkça düzlem hemen büyümektedir. Düşük karmaşıklıkları nedeniyle PLA ve PAL'lere basit programlanabilir mantıksal devre (simple programmable logic devices, SPLD) adı verilmektedir.

Karmaşık Programlanabilir Mantıksal Devre

Bir önceki kısımda PLA ve PAL'ler küçük boyutlarda kullanılabilir, birkaç yüz mantıksal kapıdan oluşur. Büyük mantıksal devreler için karmaşık programlanabilir mantıksal devre (complex programmable logic devices, CPLD) kullanılabilir.

Bir CPLD, makro hücrelerden, giriş/çıkış bloklarından ve arabağlantı ağından oluşur. Giriş/çıkış bloklarının birbirleriyle ve makro hücreler ile, makro hücrelerin birbirleri ile olan bağlantıları programlanabilir arabağlantı ağı (programmable interconnection network) ile yapılır.

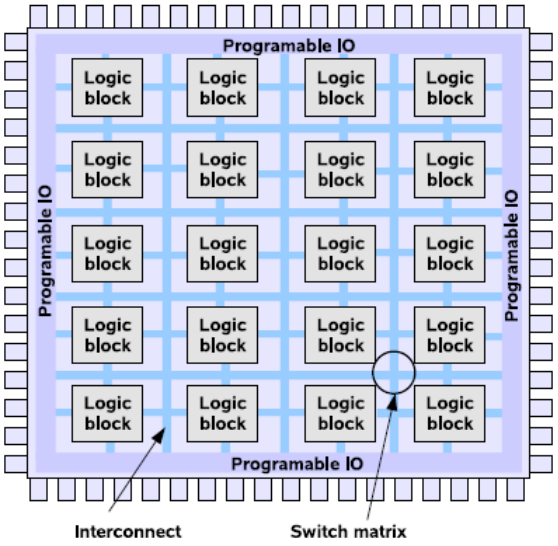
Bir makro hücre genellikle, bir kaç PLA ve flip-flop'lardan oluşur. PLA ile karşılaştırıldığında geniş kapasitesili olmasına rağmen (birkaç bin mantıksal kapı) , CPLD'ler tekrar-düzenlenebilir hesaplama donanımında kullanmak için hala küçük sayılır. CPLD'ler genellikle küçük fonksiyonların gerçekleştiriminde kullanılırlar. CPLD'ler, birçok tekrar-düzenlenebilir donanımda başlangıç konfigürasyonunda (start up configuration) kullanılır.



Şekil 11 : CPLD genel yapısı

Field Programmable Gate Array (FPGA)

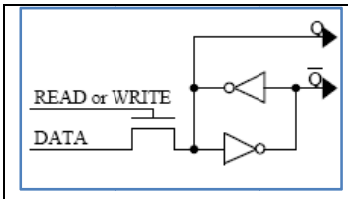
1985 yılında Xilinx şirketi tarafından tanıtılan, elektriksel sinyaller vasıtasıyla programlanıp, hızlı şekilde uyarlanabilen FPGA, CPLD'ler gibi üç ana bölümden oluşur: programlanabilir mantıksal hücreler (programmable logic cells, logic blocks), programlanabilir arabağlantı ağı (programmable interconnection network) ve girdi/çıktı hücreler. FPGA'in bu üç temel bileşeni (logic block, interconnection ve input/output) kullanıcı tarafından bir (veya birden fazla) kez programlanabilir. Şekil 13'de genel bir FPGA'in yapısı gösterilmektedir.



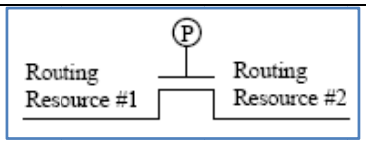
Şekil 12: FPGA genel yapısı

Yapısal olarak FPGA'ler, PAL ve MPGA (Mask-Programmable Gate Arrays) arasında oluşan melez (hybrid) bir yaklaşım olarak ortaya çıkmıştır. Programlanabilir dizi mantıklarında (PAL) olduğu gibi FPGA'ler de tamamen elektriksel sinyaller vasıtasıyla programlanabilir, hızlı şekilde uyarlanabilir. Sahip olduğu milyonlarca mantıksal kapılar sayesinde karmaşık hesaplamaları tek bir yongada yapabilmesi ile de MPGA'e benzerlik göstermektedir. Bu özellikleri ile FPGA'ler hızlı prototipleme için çok yaygın şekilde kullanılırlar. FPGA'ler sahip olduğu esneklik, kapasite ve başarım sayesinde tekrar-düzenlenebilir mimarilerin temel taşıdır.

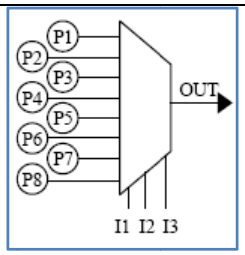
Çoğu FPGA SRAM-programlanabilirdir. FPGA konfigürasyon noktalarına bağlanan SRAM bitlerinin programlanması ile FPGA yapılandırılır. Standart RAM'in kolayca programlanabildiği gibi, FPGA yongaları da bu sayede tekrar tekrar programlanabilir. FPGA'de dağıtım yollarını uyarlamak için passgate yapıları kullanılmaktadır. Şekilde görüldüğü gibi, programlama bitine true değeri verilerek yönlendirme açılır, sayısal siyalin bir noktadan diğerine iletimi sağlanmış olur, ikili değere false verilerek aradaki bağlantı koparılır. Bu şekilde milyonlarca ara bağlantı elemanı düşünüldüğünde, sayısal sinyallerin yönlendirimi için milyonlarca seçenek elde edilmektedir. Yönlendirme elemanları istenildiği şekilde ayarlanarak verimli bir cihaz elde edilmiş olur.



Şekil 13 : SRAM-temelli FPGA için programlanabilir bit



Şekil 14 : programlanabilir routing connection



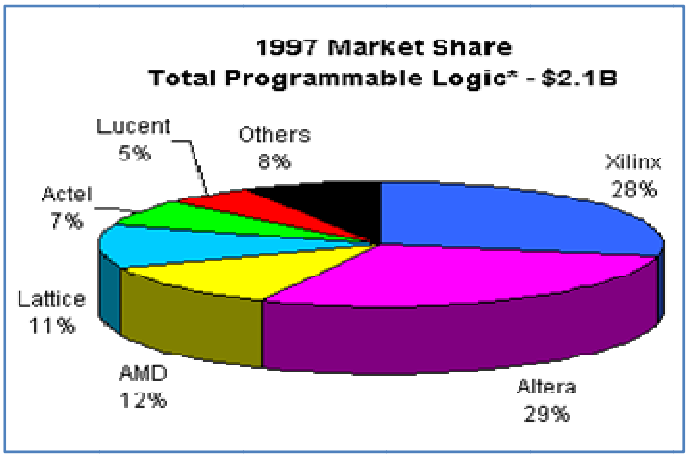
Şekil 15 : 3-girişli LUT

FPGA'lerde göreli mantıksal fonksiyonların (arbitrary logic functions) hesaplanmasında taramalı tablodan (Look Up Table, LUT) yararlanılır. LUT elemanları istenilen fonksiyon için N adet girdi için 2^N adet seçenek çıkış seçeneği sunar (yukarıdaki şekilde sağda gösterildiği gibi). Bu sayede taramalı tablolar (LUT), AND veya NAND mantıksal kapıları gibi davranması sağlanabilmektedir. FPGA'de bulunan taramalı tablolar (LUT) ile herhangi bir fonksiyon gerçeklenebilmektedir. Flip-flop pipeline yapısı için kullanılabilmektedir.

Günümüz FPGA çalışmalarında genel eğilim, geniş veri yolları kullanarak FPGA'e giden yolun bant genişliğini artırmak, veya FPGA üzerinde birden fazla konfigürasyon depolayarak konfigürasyonlar arasında kolay geçişi sağlamayı hedeflemektedir.

Sayısal sinyal işleme (DSP), uzay ve savunma sistemleri, computer vision, ses tanıma, kriptografi, biyoenformatik gibi yüksek başarımı hedefleyen uygulamalarda FPGA'lerin kullanımı oldukça yaygındır. FPGA'lerin yüksek başarımı hedefleyen uygulamalarda kullanımı günden güne artmaktadır.

Sonuç olarak tekrar-düzenlenebilir donanım pazarında yaygın olarak FPGA'ler kullanılmaktadır, Xilinx ve Altera pazarda liderdir (Şekil 17). İri-tanecikli (coarse-grained) tekrar-düzenlenebilir donanımlar konusunda farklı yaklaşımlar ve prototipler olmasına karşın pazarda henüz yer bulabilmiş değildir. Bu konuda yapılan araştırma ve geliştirmeler hızlı bir şekilde ilerlemektedir. FPGA'lerdeki tekrar-düzenlenebilirliği sağlayan temel mantık gelecekte de aynı kalacaktır.



Şekil 16 : Tekrar-düzenlenebilir donanım pazarında durum

Yukarıda grafikte 1997 yılında yeniden düzenlenebilir mimarili sistemlerin pazar payları gösterilmiştir. Günümüzde ise Xilinx ve Altera'nın pazardaki üstünlüğü devam etmektedir.

GERÇEKLEŞTİRİM

Uygulamaya göre, hesaplama için gerekli kaynaklar çalışma zamanında bileşenler şeklinde oluşturularak donanımın üzerine yüklenir. Bileşenlerin oluşturulması işine mantıksal birleşim (logic synthesis) adı verilir. mantıksal birleşim işlemi bir optimizasyon işlemidir. Amaç, en hızlı donanımı oluşturmak, en az güç tüketimi ve en az sayıda sayısal elemanı kullanarak sistemin genel maliyetini düşürmektir. Uygulamanın FPGA kaynaklarına eşlenmesi işi mantıksal birleşimin bir adımıdır. FPGA kaynaklarının uygulamaya eşlenmesi, taramalı tablonun konfigüre edilmesi ile sağlanır.

Bazı tekrar-düzenlenebilir mimarili donanımlarda, donanımın bir bölümü çalışmaya devam ederken, sadece bir bölümü üzerinde düzenleme yapılabilir. Buna parçalı tekrar-düzenleme (partial reconfiguration) adı verilmektedir. Bu sayede donanım üzerinde birçok geçici fonksiyon gerçeklenebilir.

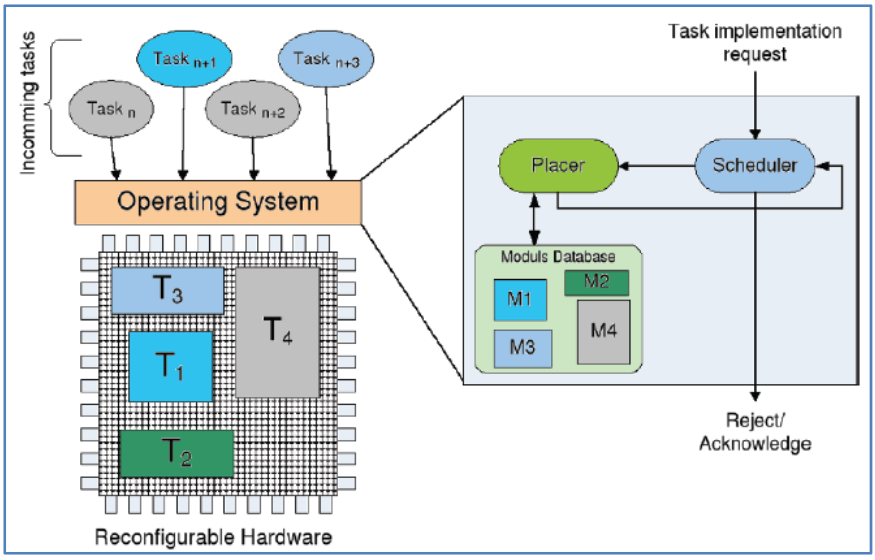
Konfigürasyonun yapılış zamanına göre tekrar-düzenleme ikiye ayrılmaktadır:

Derleme zamanı tekrar-düzenleme (Compile-time reconfiguration):

Hesaplamalar ve konfigürasyonun yüklenmesi işlemi donanımın üzerinde çalışacak uygulamanın derleme işlemi sırasında belirlenir. Çalışma zamanında donanımın tekrar-düzenlenmesi söz konusu değildir. Bu yaklaşım daha çok tamamen tekrar-yapılandırılabilir (fully reconfigurable) donanımlar için uygundur.

Çalışma zamanı tekrar-düzenleme (Run-time reconfiguration):

Derleme zamanında konfigürasyonun nasıl yapılacağı, hesaplamaların nasıl olacağı konusunda bilgi sahibi değildir. Çalışma zamanında gelen görev isteği, dinamik olarak ele alınır. Değişen koşullara ve işletim isteklerine göre donanım kendisini değişirebilmelidir. Bu bazı zorlukları sebebiyle yan etkilerinin de giderilmesi gerekmektedir. Birleştirme (defragmentation) sorunun giderilmesi ve çalışma zamanında yeni gelen modüller arasındaki iletişimin sağlanması gerekmektedir. Tekrar-düzenlenebilir donanımın yönetimi schedular ve placer vasıtasıyla sağlanır (şekil x) . "Placer" işlemci üzerinde koşan işletim sisteminin bir parçasıdır. İşlemci tekrar-düzenlenebilir yonga içerisinde veya yonganın dışında bulunabilir.

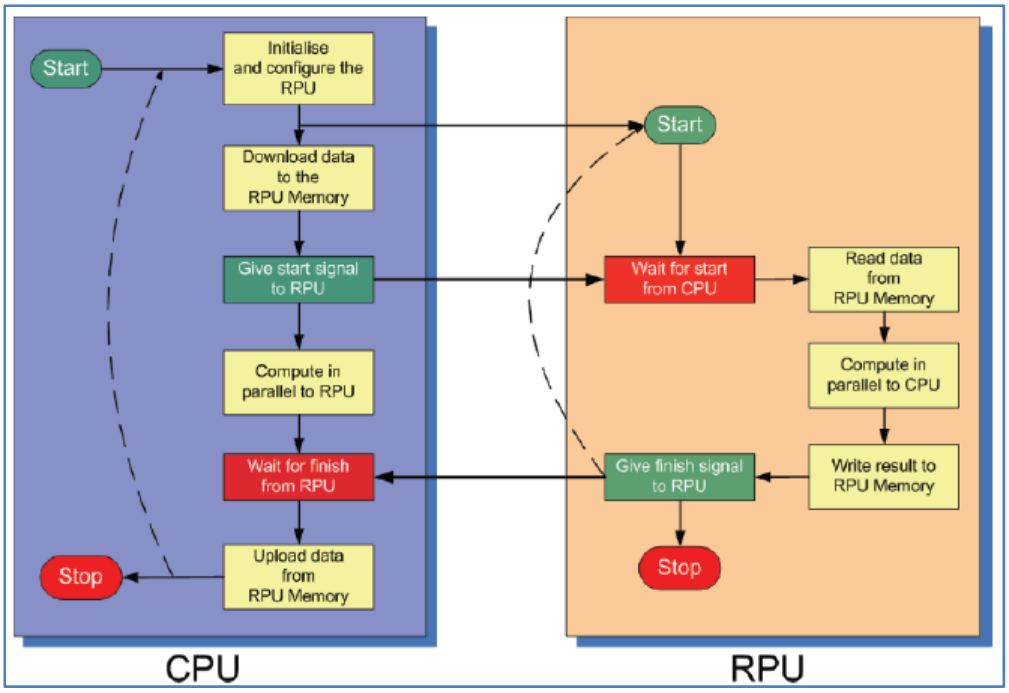


Şekil 17 : Çalışma zamanı tekrar-düzenlenebilir mimari

İşdüzenleyici (scheduler) görevlerin ne zaman ve nasıl yürütüleceğine karar verir. Veri tabanındaki konfigürasyon bilgisi niteliğindeki görevler, görevin bounding box'ı görevin çalışma zamanına göre tanımlanırlar. Bounding box, görevin donanımın hangi kısmında işletileceğini tanımlar. İşdüzenleyici (scheduler) aynı zamanda hangi görevlerin RPU'da işletileceğine karar verdikten sonra bu görevi placer'a devreder, bu aşamadan sonra placer görevi donanım üzerinde işletmekten sorumludur. Placer görevi işletmek için uygun zeminin oluşmaması durumunda söz konusu görevi işdüzenleyiciye geri gönderir. İşdüzenleyici daha sonra aynı görevi, tekrardan gönderebilir ya da placer'a yeni bir görev verebilir. Böyle bir durumda ilk verilen görev reddedilmiş olur.

İster derleme zamanı tekrar-düzenleme yöntemi, isterse çalışma zamanı tekrar-düzenleme yöntemli olsun, hesaplama ve tekrar-düzenleme akışı aşağıda şekil 18'de gösterildiği gibidir. Merkezi işlemci (host CPU) donanımın tekrar yapılandırılmasından ve veri transfer işlemlerinden sorumludur. Genellikle tekrar-düzenlenebilir birim (RPU) ile merkezi işlemcinin iletişimi veriyolu (bus) üzerinden sağlanır. RPU, merkezi işlemci birimi tarafından çağrılabilen komut seti yardımcı işlemci (coprocessor) rolünü üstlenir. Hesaplama akışı aşağıdaki algoritmada gösterildiği şekildedir.

```
Start
Initialize the RPU
while (1) do
      Configure the RPU to implement a new task
      Download Data for RPU computation into RPU-memory
      Computes in parallel with the RPU if necessary
      Upload the data computed by the RPU from the RPU-memory
end while
Stop
```



Şekil 18 : CPU-RPU konfigürasyon ve hesaplama adımları

REFERANSLAR

[1] Christophe Bobda, Introduction to Reconfigurable Computing, Architectures, Algorithms, and Applications, , University of Kaiserslautern, Germany

[2] Reconfigurable Computer Architectures, Marco Platzner, Computer Systems Laboratory, Stanford University

[3] Reconfigurable Computing Systems Cost/Benefit Analysis Model, William W.C Chu, A thesis presented to the University of Waterloo, Electrical and Computer Engineering, Waterloo, Ontario, Canada, 2005

[4] MorphoSys: A Reconfigurable Architecture for Multimedia Applications, Hartej Singh, Ming-Hau Lee, Guangming Lu, Fadi J. Kurdahi, Nader Bagherzadeh, University of California, Irvine, CA (United States), and Eliseu M.C. Filho, Federal University of Rio de Janeiro (Brazil)

[5] Reconfigurable Computing, Tony Givargis and Nikil Dutt