Mühendislik Projesi
Engineering Project

Hazırlayan: M. Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Bu dersin sunumları, “Ralph M. Ford, Chris S. Coulston, Design for Electrical and Computer Engineers, McGraw Hill, 2008.” kitabı kullanılarak hazırlanmıştır.

İçerik

- Bottom-up ve top-down tasarım
- Fonksiyonel ayrıştırma
- Sayısal tasarım uygulaması
- Yazılım tasarımı uygulaması
- Bağlama ve birleştirme

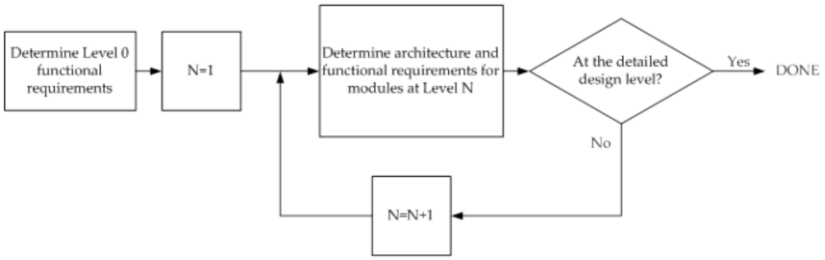
Bottom-up ve top-down tasarım

- Kavram geliştirme tamamlandıktan sonra, **sistem gereksinimlerini karşılayacak çözüm oluşturulur.**
- **Fonksiyonel ayrıştırmada, sistem iteratif olarak alt sistem bileşenlerine ayrıştırılır.**
- Mühendislik tasarımında iki genel yaklaşım vardır:
  - Bottom-up
  - Top-down
- **Bottom-up** (aşağıdan yukarıya) yaklaşımda, **sistemin temel bileşenlerinden başlanır** ve tüm sistem oluşturulur.
- **Top-down** (yukarıdan aşağıya) yaklaşımda, **tasarımcı son sistemin ne olduğu görüşüne sahiptir** ve problem alt parçalara ayrılır.
- **Günümüzdeki tasarımlarda iki yaklaşım birlikte kullanılabilmektedir.**

İçerik

Fonksiyonel ayrıştırma

- Fonksiyonel ayrıştırma, **iteratif bir süreçtir ve sistemin tüm bileşenlerinin işlevi tanımlanır.**
- İteratif ayrıştırma **fiziksel bileşenlere ulaşıncaya kadar devam eder.**



- Tasarım en üst seviyeden (Level 0) başlar ve **sistem için fonksiyonel gereksinimler belirlenir.**
- Level 1 (design architecture) ile **sistemin mimari tasarımı** ve **modüller arasındaki bağlantılar belirlenir.**
- **Fiziksel bileşenlere** (devre elemanları, kapı devreleri, …) **ulaşınca** (detailed design) **ayrıştırma biter.**

Fonksiyonel ayrıştırma

- Fonksiyonel ayrıştırmada, aynı seviyedeki **modüller aynı karmaşıklık düzeyinde olmalıdır.**
- **Tasarlanan modüller** birlikte çalışmak için **ara yüzlere sahip olmalıdır.**
- **Top-down** yaklaşım **dikey düşünme sürecini izler** ve tasarımcı problemden **çözüme doğrusal olarak ilerler.**
- **Yanal düşünme** yaklaşımı da uygulanarak **yenilikçi çözümler geliştirilmelidir.**
- **Fonksiyonel ayrıştırma, tüm sistem davranışını tanımlamaz.**
- **Akış diyagramları**, **durum diyagramları** ve **veri akış diyagramları** da kullanılmalıdır.
- Mevcut ve **güncel teknoloji** kullanılmalıdır.
- Tasarım **basit** ve **sade olmalıdır, gereksiz karmaşıklık oluşturulmamalıdır.**

İçerik

- Bottom-up ve top-down tasarım
- Fonksiyonel ayrıştırma
- Sayısal tasarım uygulaması
- Yazılım tasarımı uygulaması
- Bağlama ve birleştirme

Sayısal tasarım uygulaması

- Fonksiyonel ayrıştırma sayısal tasarım uygulamarında yaygın kullanılır (entity-architecture design).
- **Giriş** ve **çıkışlar,** varlıkları (entity) ifade eder.
- **Mimari, fonksiyonelliği tanımlar.**

Örnek: kronometre tasarımı

- Aşağıdaki kısıtları ve özellikleri sağlayan bir kronometre tasarımı yapınız.
  - İkiden fazla kontrol butonu olmayacak.
  - **Run**, **stop** ve **reset** işlevlerine sahip olacak.
  - Çıkış 16-bit binary sayı olacak ve toplam geçen saniyeyi gösterecek.

Sayısal tasarım uygulaması

Örnek: kronometre tasarımı-devam

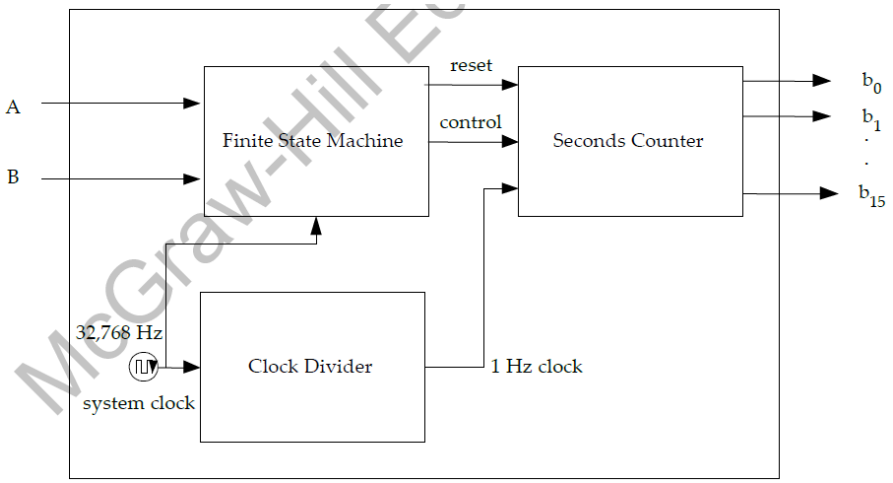
- **Level 0:** Fonksiyonel gereksinim diyagramı aşağıdaki gibidir.

Module	Stopwatch
Inputs	- A: Reset button signal. When the button is pushed it resets the counter to zero. - B: Run/stop toggle signal. When the button is pushed it toggles between run and stop modes.
Outputs	- b_{15}–b_0: 16-bit binary number that represents the number of seconds elapsed.
Functionality	The stopwatch counts the number of seconds after B is pushed when the system is in the reset or stop mode. When in run mode and B is pushed, the stopwatch stops counting. A reset button push (A) will reset the output value of the counter to zero only when the stopwatch is in stop mode.

Sayısal tasarım uygulaması

Örnek: kronometre tasarımı-devam

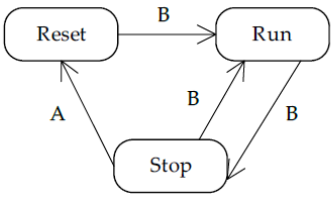
- **Level 1:** Saniye sayıcı, saat bölücü ve sonlu durum makinesinden oluşur.



Sayısal tasarım uygulaması

Örnek: kronometre tasarımı-devam

- **Level 1:** Modüllerin işlevsellikleri aşağıdaki gibi tanımlanır.
- **FSM modülü**

Module	Finite State Machine
Inputs	- A: Signal to reset the counter. - B: Signal to toggle the stopwatch between run and stop modes. - Clock: 1 Hz clock signal.
Outputs	- Reset: Signal to reset the counter to zero. - Control: Signal that enables or disables the counter.
Functionality	

Sayısal tasarım uygulaması

Örnek: kronometre tasarımı-devam

- **Level 1:** Modüllerin işlevsellikleri aşağıdaki gibi tanımlanır.
- **Saat bölücü**

Module	Clock Divider
Inputs	- System clock: 32,768 Hz.
Outputs	- Internal clock: 1 Hz clock for seconds elapsed.
Functionality	Divide the system clock by 32,768 to produce a 1 Hz clock.

- **Saniye sayıcı**

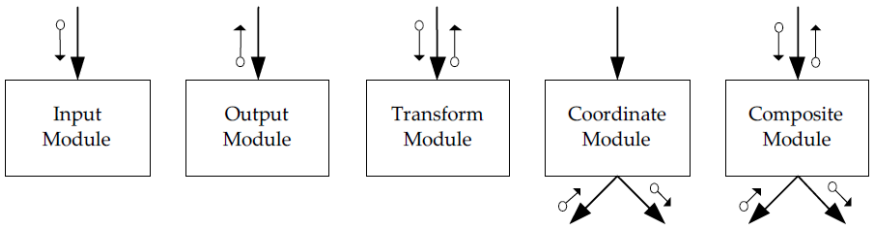
Module	Seconds Counter
Inputs	- Reset: Reset the counter to zero. - Control: Enable/disable the counter. - Clock: Increment the counter.
Outputs	- b_{15}–b_0: 16-bit binary representation of number of seconds elapsed.
Functionality	Count the seconds when enabled and resets to zero when reset signal enabled.

İçerik

- Bottom-up ve top-down tasarım
- Fonksiyonel ayrıştırma
- Sayısal tasarım uygulaması
- Yazılım tasarımı uygulaması
- Bağlama ve birleştirme

Yazılım tasarımı uygulaması

- **Programlama dilleri** fonksiyon çağırma ve **alt modül oluşturma gibi özelliklere sahiptir.**
- Fonksiyonel yazılım tasarımı, **tekrarlı fonksiyon çağırmaları** ile **fazla kod yazımı** ihtiyacını ortadan kaldırır.
- **Yapı kartları** (structure charts) fonksiyonel yazılım tasarımını görselleştirmek için kullanılan blok diyagramlardır.



- **Büyük oklar** modüller arası **bağlantıları, küçük oklar** modüller arası **veri** ve **kontrol** bilgisi **akışını gösterir.**

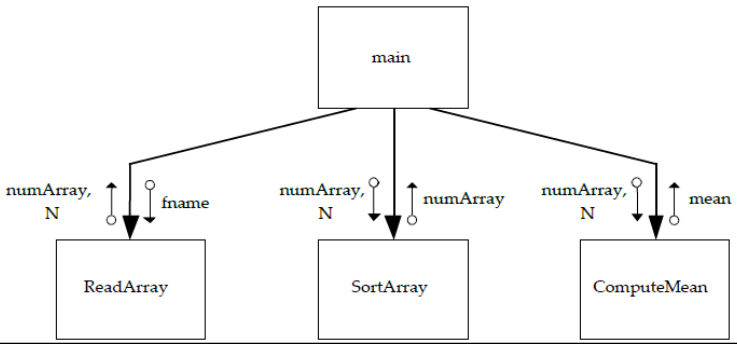
Yazılım tasarımı uygulaması

- **Yapı kartları ile 5 temel modül oluşturulur:**
  - Giriş modülleri
  - Çıkış modülleri
  - Dönüştürme modülleri
  - Koordinasyon modülleri
  - Birleşik modüller
- **Giriş modülleri**, veri alır.
- **Çıkış modülleri**, veri gönderir.
- **Dönüştürme modülleri**, veri alır, işler ve sonucu döndürür.
- **Koordinasyon modülleri**, modüller arasında senkronizasyonu ve koordinasyonu sağlar.
- **Birleşik modüller**, diğer modüllerin birleşimiyle oluşur.

Yazılım tasarımı uygulaması

Örnek: Aşağıdaki kısıtları ve özellikleri sağlayan bir yazılım tasarımı yapınız.

- Tamsayılardan oluşan ASCII dosyayı giriş olarak kabul edecek.
- Sayıları artan sırada sıralayacak.
- Sıralanmış sayıları disk üzerine saklayacak.
- Sayıların ortalamasını hesaplayacak.
- Ortalama değeri ekranda gösterecek.



İçerik

- Bottom-up ve top-down tasarım
- Fonksiyonel ayrıştırma
- Sayısal tasarım uygulaması
- Yazılım tasarımı uygulaması
- Bağlama ve birleştirme

Bağlama ve birleştirme

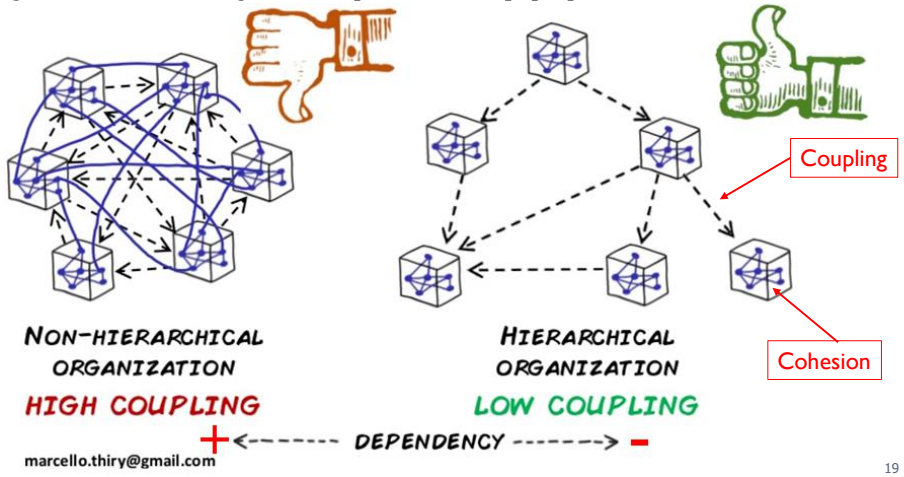
- Eğer bir sistemde **2 modül varsa en fazla 1 bağlantı olabilir.**
- Eğer 3 modül varsa en fazla 3, 4 modül varsa en fazla 6 bağlantı olabilir.
- ***n* modül için en fazla bağlantı sayısı:**

$$\text{Connections}_{\text{max}} = \frac{n(n-1)}{2}$$

- Coupling, hangi alt sistem veya modüllerin bağlanarak genişleyeceğini belirler.
- Modüller arasında **veri ve kontrol alışverişi arttıkça bağlantı (coupling) derecesi artar.**
- Bir sistemin **bağlantı derecesi artarsa** bir modülde **değişiklik yapma maliyeti artar.**

Bağlama ve birleştirme

- Cohesion, **birden fazla işlevsel bileşeni aynı modül içerisinde birleştirir.**
- **Modüller arasında bağlantı (coupling) yapılır, modül içerisinde birleştirme (cohesion) yapılır.**



Ödev

- Aşağıdaki özelliklere sahip bir otonom otomobil için fonksiyonel tasarım geliştiriniz.
  - Belirli bir güzergah üzerinde hareket edecektir.
  - Güzergah üzerinde trafik lambasını ve yol şeritlerini algılayacaktır.
  - Şerit takibi yapacaktır.
  - Yoldaki maksimum hız 55km/saat olacaktır.
  - Olabildiği kadar kısa sürede güzergahı tamamlayacaktır.