EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

AYGIT AĞACI YAZILIMLARININ

MODEL GÜDÜMLÜ GELİŞTİRİLMESİ

Sadık ARSLAN

Tez Danışmanı: Doç. Dr. Geylani KARDAŞ

Uluslararası Bilgisayar Anabilim Dalı

Sunuş Tarihi: 18.06.2018

Bornova-İZMİR

2018

Sadık ARSLAN tarafından YÜKSEK LİSANS tezi olarak sunulan "Aygıt Ağacı Yazılımlarının Model Güdümlü Geliştirilmesi" başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 18.06.2018 tarihinde yapılan tez savunma sınavında aday oybirliği ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	: Doç. Dr. Geylani KARDAŞ	.....
Raportör Üye	: Dr. Öğr. Üyesi İlker KOCABAŞ	.....
Üye	: Dr. Öğr. Üyesi Kaya OĞUZ	.....

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum "Aygıt Ağacı Yazılımlarının Model Güdümlü Geliştirilmesi" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

18 /06 /2018

İmzası

Adı-Soyadı

Sadık ARSLAN

ÖZET

AYGIT AĞACI YAZILIMLARININ MODEL GÜDÜMLÜ GELİŞTİRİLMESİ

ARSLAN, Sadık

Yüksek Lisans Tezi, Uluslararası Bilgisayar Anabilim Dalı
Tez Danışmanı: Doç. Dr. Geylani KARDAŞ
Haziran 2018, 55 sayfa

Donanım bilgisi ve yapılandırmasını içeren Aygıt Ağacı (DT) dosyaları çeşitli gömülü platformlar için işletim sistemlerinin derlenmesi sırasında kullanılırlar. DT'ler, bir gömülü sistem donanımı içindeki fiziksel aygıtların ve çevre birimlerinin açıklamasını düğüm özellikleri ile sağlar. Bununla birlikte, yazılım geliştiricileri çoğunlukla bilinen genel amaçlı programlama dillerinden farklı bir yapıya sahip olan DT kaynak dosyalarının metin tabanlı sözdizimi nedeniyle bu tür uygulamaları geliştirmede zorluk çekmektedirler. Ayrıca, geliştiriciler, bu tip DT dosyalarını hazırlamak için farklı mikroişlemcilere özgü donanımları bilmesi gerekmektedir. Bu eksiklikleri ve mevcut DT yazılım geliştirme süreçlerinin zorluklarını ortadan kaldırma amacıyla, bu tezde farklı gömülü sistem platformları için DT yazılımının otomatik oluşturulmasını ve yapılandırılmasını sağlayan model-güdümlü bir yazılım geliştirme yöntemi sunulmaktadır. Ayrıca, önerilen yöntemin uygulanması için projedeki tüm destekleyici görsel modelleme ve otomatik kod oluşturma araçlarını içeren DSML4DT adlı bir alana özgü modelleme dili geliştirilmiştir. Değerlendirme için bir gömülü sistem cihazı kullanılmıştır. Karşılaştırmalı değerlendirme sonuçları, DT yazılımının bu dili kullanarak gerekli kodun önemli bir miktarını otomatik olarak sağlayabileceğini göstermiştir.

Anahtar Sözcükler: Model-Güdümlü Yazılım Geliştirme, Model-Güdümlü Mimari, Alana-Özgü Modelleme Dili, Aygıt Ağacı, Gömülü Yazılım.

ABSTRACT

MODEL-DRIVEN DEVELOPMENT OF DEVICE TREE SOFTWARE

ARSLAN, Sadık

M.Sc. in International Computer
Supervisor: Assoc. Prof. Dr. Geylani KARDAŞ
June 2018, 55 pages

Device Tree (DT) files, which include hardware information and configuration, are used during the compilation of operating systems for various embedded platforms. DTs provide description of physical devices and peripherals inside an embedded system hardware with node specifications. However, software developers mostly have difficulties in developing such applications due to text-based syntax of DT source files which has a different structure from the well-known general purpose programming languages. Furthermore, the developer needs to be familiar with the hardware which is specific for each different microprocessor to prepare such DT files. In order to eliminate these deficiencies and difficulties of current DT software development processes, a model-driven software development methodology in which automatic generation and configuration of DT software for different embedded system platforms is provided in this thesis. Also, a domain-specific modeling language, called DSML4DT, is developed with including all supporting visual modeling and automatic code generation tools for the application of the proposed methodology. An embedded system device is used for evaluation. The comparative evaluation results showed that DT software can automatically provide a significant amount of the required code using DSML4DT.

Keywords: Model-Driven Software Development, Model-Driven Architecture, Domain-specific Modeling Language, Device Tree, Embedded Software.

TEŞEKKÜR

Yüksek lisans eğitimim sürecinde ve tez çalışmam sırasında bilgi ve tecrübeleriyle bana yol gösteren, desteğini esirgemeyen danışman hocam sayın Doç. Dr. Geylani KARDAŞ'a çok teşekkür ederim.

Eğitim hayatım boyunca bana sürekli destek olan aileme teşekkürü borç bilirim.

Ayrıca desteklerinden dolayı Kent Kart Ege Elektronik A.Ş.'ye ve Kent Kart Ege Elektronik A.Ş. Araştırma Geliştirme Bölümü çalışanlarına çok teşekkür ederim.

Bu tez çalışmasının bir bölümü 117E553 no'lu ve "Aygıt Ağacı Yazılımlarının Farklı Gömülü Sistem Platformları için Model Güdümlü Geliştirilmesi" başlıklı TÜBİTAK projesi kapsamında gerçekleştirilmiştir.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

ÇİZELGELER DİZİNİ

Çizelge Sayfa

LİSTELER DİZİNİ

Liste Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama
ARM	Acorn RISC Makinesi (ing. Acorn RISC Machine)
CAAM	Kriptografik Hızlanma ve Güvence Modülü (ing. Cryptographic Acceleration and Assurance Module)
CAD	Bilgisayar Destekli Tasarım (ing. Computer Aided Design)
CAN	Kullanıcı Alan Ağı (ing. Controller Area Network)
CPU	Merkezi İşlem Birimi (ing. Central Processing Unit)
DSL	Alana-Özgü Dil (ing. Domain-specific Language)
DSML	Alana-Özgü Modelleme Dili (ing. Domain-specific Modeling Language)
DSML4DT	Aygıt Ağaçları için Alana-Özgü Modelleme Dili (ing. Domain Specific Modeling Language for Device Trees)
DT	Aygıt Ağacı (ing. Device Tree)
EEPROM	Elektronik Olarak Silinebilir Sadece Okunur Bellek (ing. Electronically Erasable Programmable Read-Only Memory)
ESAI	Geliştirilmiş Seri Ses Arabirimi (ing. Enhanced Serial Audio Interface)
FPGA	Alan Programlanabilir Kapı Dizileri (ing. Field Programmable Gate Arrays)
GME	Grafik Modelleme Ortamı (ing. Graphical Modeling Environment)
GPIO	Genel-Amaçlı Giriş/Çıkış (ing. General-Purpose Input/Output)

SİMGELER VE KISALTMALAR DİZİNİ (devam)

Kısaltmalar	Açıklama
GPS	Küresel Konumlandırma Sistemi (ing. Global Positioning System)
GPU	Grafik İşleme Birimi (ing. Graphical Processing Unit)
GSM	Mobil İletişim İçin Küresel Sistem (ing. Global System for Mobile Communications)
HDMI	Yüksek Çözünürlüklü Multimedya Arayüzü (ing. High Definition Multimedia Interface)
I2C	Entegreler Arası Devre (ing. Inter-Integrated Circuit)
I2S	Entegreler-Arası Ses (ing. Inter-IC Sound)
IPU	Görüntü İşleme Birimi (ing. Image Processing Unit)
LCD	Likit Kristal Ekran (ing. Liquid Cyristal Display)
LDB	LVDS Görüntü Köprüsü (ing. LVDS Display Bridge)
LoC	Kod Satır Sayısı (ing. Lines of Codes)
LVDS	Düşük Gerilimli Diferansiyel Sinyal (ing. LVDS Display Bridge)
M2T	Model-Metin (ing. Model-to-Text)
MDA	Model-Güdümlü Mimari (ing. Model-Driven Engineering)
MDD	Model-Güdümlü Geliştirme (ing. Model-Driven Development)
MDE	Model-Güdümlü Mühendislik (ing. Model-Driven Engineering)

SİMGELER VE KISALTMALAR DİZİNİ (devam)

Kısaltmalar	Açıklama
MDSD	Model-Güdümlü Yazılım Geliştirme (ing. Model-Driven Engineering)
MIMO	Çok-Giriş Çok-Çıkış (ing. Multiple-Input Multiple-Output)
MMC	Multimedia Kart (ing. Multimedia Card)
MOF	Meta-Nesne Tesisi (ing. Meta-Object Facility)
MTL	Model-metin Dönüşüm Dili (ing. Model to Text Language)
NAND	Negatif-ve Kapısı (ing. negative-AND)
OCL	Nesne Kısıtlama Dili (ing. Object Constraint Language)
OMG	Object Management Group
PCI	Çevresel Komponent Arabağlantı (ing. Peripheral Component Interconnect)
PCI-E	Hızlı PCI (ing. PCI-Express)
PIM	Platform Bağımsız Model (ing. Platform Independent Model)
PSM	Platform Spesifik Model (ing. Platform Specific Model)
PWM	Darbe-Genişlik Modülasyonu (ing. Pulse-Width Modulation)
RAM	Rasgele Erişimli Bellek (ing. Random Access Memory)
RGB	Kırmızı Yeşil Mavi (ing. Red Green Blue)
RF	Radyo Frekansı (ing. Radio Frequency)

SİMGELER VE KISALTMALAR DİZİNİ (devam)

Kısaltmalar	Açıklama
RTC	Gerçek Zamanlı Saat (ing. Real Time Clock)
SAM	Güvenli Erişim Modülü (ing. Secure Access Module)
SD	Güvenli Sayısal Hat (ing. Secure Digital)
SDIO	Güvenli Sayısal Giriş/Çikış (ing. Secure Digital Input/Output)
SDMA	Akıllı Doğrudan Bellek Erişimi (ing. Smart Direct Memory Access)
SoC	Yonga Üstü Sistem (ing. System on Chip)
SPBA	Paylaşımlı Çevresel Hat Arayüzü (ing. Shared Peripherals Bus Interface)
SPDIF	Sony/Phillips Sayısal Arayüzü (ing. Sony/Phillips Digital Interface)
SPI	Seri Çevresel Arayüz (ing. Serial Peripheral Interface)
SSI	Seri Ses Arayüzü (ing. Serial Sound Interface)
UML	Birleşik Modelleme Dili (ing. Unified Modeling Language)
USB	Evrensel Seri Veriyolu (ing. Universal Serial Bus)
XMI	XML Metaveri Değişimi (ing. XML Metadata Interchange)
XML	Genişletilebilir İşaretleme Dili (ing. eXtensible Markup Language)

1. GİRİŞ

Gömülü sistemler, kişisel bilgisayarlardan farklı olarak, daha önce özel olarak tanımlanmış görevleri gerçekleştirmektedirler (Kamal, 2009; Lee and Seshia, 2016). Gömülü sistemler genellikle, sınırlı güçte bir işlemci, bellek ve diğer çevre birimlerine sahiptir vebu sistemlerde Embedded Linux, Android, JavaOS, LynxOS, Mobilinux ve Windows CE gibi birçok işletim sistemi kullanılmaktadır. Cep telefonları, araç takip cihazları, ağ cihazları, motor kontrol cihazları, fren sistemleri, ev otomasyon ürünleri, hava savunma sistemleri, medikal ürünler ve ölçüm sistemleri gömülü sistemlere örnektir.

Genel olarak, gömülü sistemler bir işlemci, güç birimleri, bellek, kayıt birimleri, iletişim ara yüzleri, algılayıcılar gibi birimlere sahiptir (Gajski et al., 1994). Günümüzde, sürekli gelişen sistemlerde birçok farklı donanım çevre birimi kullanılmaktadır. Ayrıca, işletim sistemleri ve işletim sistemi çekirdekleri de sık sık değişmektedir. Birçok farklı kuruluş ve şirket, Yonga Üstü Sistem (ing. System on Chip) (SoC) platformlarında farklı sistemler için işletim sistemi dağıtımları yayınlamaktadır.

Bir Aygıt Ağacı (ing. Device Tree) (DT), gömülü sistem donanımının fiziksel aygıt bileşenlerinin düğümlerle tanımlanmasını sağlayan bir veri yapısıdır (Petazonni, 2013; Simmonds, 2015). Ayrıca "Open Firmware" ve "Power Architecture Platform Requirements" gibi standartlar dâhilinde kullanılan DT, gömülü sistemdeki donanım bileşenlerinin eksiksiz bir teknik tanımını sağlamaktadır (Devicetree Community, 2016). Bir işletim sistemi çekirdeği derlendikten sonra, farklı donanım yapılandırmaları için düzenlenmiş, farklı ağaç yapılarına sahip, büyük işlemci aileleri için destekleri içermektedir. Linux işletim sistemi çekirdeği ile kullanılan DT, Acorn RISC Machine (ARM), x86, MicroBlaze, PowerPC ve SPARC işlemci mimarileri ile çalışabilmektedir. Özellikle ARM işlemcileri için, birçok şirket çok sayıda mikro işlemci ürününe sahiptir. Bu nedenle, DT kaynak dosya kullanımı ARM platformları için çok önemlidir (Devicetree Community, 2016).

Öte yandan, işletim sistemi çekirdeklerinde, tüm çevre birimlerinin sürücü dosyaları çekirdek kaynak kodunda bulunur. DT dosya desteğine sahip olmayan çekirdeklerde, saat sinyali frekansı, bacak adı ve kesmeler gibi donanım bilgilerini değiştirmek için çekirdek kodunun değiştirilmesi gerekir. Bu sürecin hem bakım maliyeti yüksektir hem de uzun zaman almaktadır. Bu problemleri çözmek için DT

yaklaşımı geliştirilmiştir. İşletim sistemi çekirdeğinde bulunan donanım özellikleri, derlenmiş bir aygıt ağacı yapısına dönüştürülür ve çekirdekten dışa aktarılır. Çekirdek derlendiğinde, donanım sisteminin özellikleri hariç tutulur ve genel amaçlı bir işletim sistemi çekirdeği elde edilir. DT yapısı, çevresel işlemlerin çekirdek kaynak koduna dokunmadan gerçekleştirilmesini sağlar. Ancak, DT yapısını özellikle çok sayıda farklı mikroişlemci mimarisinde çalışacak sistemlerin geliştirilmesinde uygulamak zordur. Bu zorluklar, geleneksel programlama tekniğinden farklı bir yapıya sahip olan DT'nin, geliştiriciler tarafından öğrenilmesi ve farklı platformlar için aynı DT yapılandırmasını tekrarlamak olarak listelenebilir.

Gömülü sistem yazılım geliştiricileri genellikle metin tabanlı ve mevcut programlama dillerinin sözdiziminden farklı olan DT kaynak dosyalarını kullanmakta güçlük çekmektedirler. DT çalışmaları için kullanılacak her yeni platforma özel metin dosyasının ayrı olarak ve en baştan hazırlanması gerekmektedir. Hazırlanan dosyalarda, sistemde kullanılan donanım parametrelerine göre bloklar sözdizimine uygun olarak kodlanır. Ağaç yapısı, DT dosyası hazırlama sırasında da dikkate alınır. Ek olarak, geliştiricinin mikroişlemciye özgü donanımı iyi tanıması gerekmektedir. Bu dosyaları hazırlamak, ilgili alan hakkında bilgi sahibi olan ancak yazılım geliştirme konusunda çok az bilgi ve deneyime sahip geliştiriciler için zordur. Ayrıca, farklı mikroişlemci mimarileri için bazı DT bileşenlerinin kodlanması ve/veya yapılandırılması, birçok geliştirici için zor ve zaman alıcıdır.

DT yapısı için yeni bir yazılım geliştirme yöntemine ve yukarıda bahsedilen problemlerin çözümü için yazılım geliştirme araçlarına ihtiyaç vardır. Bu eksikliklere dayanarak, bu tezde gömülü sistemlerde kullanılan DT yazılımlarının model-güdümlü geliştirilmesi için yeni bir yazılım geliştirme yöntemi geliştirilmiştir. Bu yöntemi uygulamak için de yeni bir Alana-Özgü Modelleme Dili (ing. Domain-specific Modeling Language) (DSML) oluşturulmuştur.

Bu tezde, hem sözdizimi hem de semantik tanımlarıyla DT'ler için bir DSML önerilmektedir. DSML, uluslararası "Devicetree" tanımı ve standardına (Devicetree Community, 2016) uygun olarak gömülü sistem DT yazılımının geliştirilmesini sağlanmaktadır. DT'lerin Model-güdümlü Mühendislik'e (ing. Model-Driven Engineering) (MDE) dayalı geliştirilmesi sırasında dil ve grafiksel araçların nasıl kullanılabileceği tezde gösterilmiştir. Hazırlanan bu DSML, Aygıt Ağaçları için Alana-Özgü Modelleme Dili (ing. Domain Specific Modeling Language for Device

Trees) (DSML4DT) olarak isimlendirilmiştir. DT uzmanlarını kendi sistemlerini programlarken desteklemek ve görsel olarak ince ayar yapabilmek için, DSML4DT bir aygıt ağacı organizasyonunun tüm yönlerini kapsamaktadır. Bir DT'nin bu klasik bakış açılarına ek olarak, DSML4DT, özellikle DT ortamının geliştirilmesine yönelik yeni bakış açıları da içermektedir. Tezde aynı zamanda DSML4DT'nin kullanımının bir değerlendirilmesi de yapılmış; doğrudan DT kodlamaya göre DSML4DT'nin kullanımının sağladığı avantajlar belirlenmiştir.

Tezin kalan kısmı şu şekilde organize edilmiştir: Bölüm 2'de, tezin altyapısını oluşturan DT alanı, model-güdümlü geliştirme ve alana-özgü dillerhakkında önbilgiler verilmiş; tezde kullanılan araçlar ve yöntemler tanıtılmıştır. Bölüm 3'te, DT ve model-güdümlü uygulamaların literatür incelemesi yapılmıştır. Tez kapsamında geliştirilen dilin soyut sözdizimi 4. Bölüm'de verilmiştir. Bölüm 5'te, DSML4DT somut sözdizimi tanıtılmaktadır. Bölüm 6'da, model-kod dönüşümü üzerinden sağlanan DSML4DT işletimsel semantiği anlatılmıştır. Bölüm 7'de, geliştirilen dil için yapılmış bir örnek vaka çalışması bulunmaktadır. 8. Bölüm'de dilin değerlendirilmesi vaka analizleri üzerinden yapılmıştır. 9. Bölüm'de ise tez ile ilgili sonuç ve ileriye yönelik çalışma önerileri yer almaktadır.

2. ÖNBİLGİLER

2.1 Aygıt Ağacı Yapısı

Basit olarak söylemek gerekirse, DT, donanımın yapılandırmasını açıklayan bir veri yapısıdır. Bu yapı, çalışılan gömülü sistemin işlemcisi, belleği, veri yolları ve çevre birimleri gibi birçok blüm hakkında bilgi içerir. İşletim sistemi, önyükleme (bootloading) sırasında DT yapısını ayrıştırır ve mikroişlemciyi nasıl yapılandıracağını burada belirler. Ayrıca yüklenecek aygıt sürücüleri hakkında kararlar almak için DT yapısı kullanılır.

DT yapısı, "/" karakteri ile gösterilen, kök olarak adlandırılan bir düğümle başlamaktadır. DT'de her düğümden çok sayıda, ismi olan, numarası olabilen birden fazla çocuk düğüm oluşturulabilmektedir. Düğümler isteğe göre, ek veri içeren nitelik değerlerini içerebilmektedirler. DT yapısı, IEEE 1275-1994 standardı tarafından daha önceden belirlenmiş olan kurallara uygun olarak tasarlanmaktadır (IEEE Standard 1275-1994, 1994).

DT yapısının kendine özgü bir sözdizimi vardır. (Arslan vd., 2017) çalışmasında bir DT mimarisinde bir örnek kullanım tanımlanmıştır. Aygıt Ağacı Kaynağı (ing. Device Tree Source, .dts) dosya formatı, aygıt ağaçlarını yazılım geliştiricileri tarafından düzenlenebilecek biçimde ifade etmek için kullanılır. Aygıt Ağacı Derleyici Aracı (ing. Device Tree Compiler Tool, dtc), DT'nin .dts biçimini, işletim sistemlerinin gerektirdiği İkili Aygıt Ağacı (ing. Binary Device Tree Blob, .dtb) biçimine dönüştürmek için kullanılır. DT yapısında, genel bir ağaç yapısı gibi ebeveyn ve çocuk düğümleri vardır. Örnek olarak, basit bir kök düğümü ve alt düğümleri içeren .dts dosyası Liste 2.1'de görülebilir.

```
01  / {                                          // kök düğümü
02      bir-nitelik;                             //bir donanım özelliği
03      bir-cocuk-dugum {                                //ilk çocuk düğüm
04        dizi-niteligi = <0x10 53>;
05        yazi-niteligi = "merhaba, dunya";
06      };
07      diger-cocuk-dugum {                      //diğer çocuk düğüm
08        binary-nitelik = [0221ALI];
09        yazi-listesi = "evet","hayir","belki";
10      };
11  };
```

Liste 2.1 DT yapısı örneği

2.2 Model-Güdümlü Mühendislik ve Alana-Özgü Diller

Model-Güdümlü Mühendislik, yazılım ürünlerini tasarlamak ve geliştirmek için Model-Güdümlü Geliştirme (ing. Model-Driven Development) (MDD) paradigmasını (Selic, 2003) kullanmaktadır. MDD paradigması, bilgisayar programlarını hızlı, etkin ve minimum maliyetle geliştirmeyi amaçlamaktadır.

Model-Güdümlü Mimari (ing. Model-Driven Engineering) (MDA), MDD'nin ya da MDE'nin yazılım geliştirmeye nasıl uygulanabileceğini açıklayan ve yaygın olarak kullanılan bir standarttır (OMG, 2003). MDA'da çeşitli yazılım metamodel ve modelleme seviyeleri tanımlanmıştır. Modeller MDE bakış açısından tanımlanmıştır (Schmidt, 2003). MDA'de modeller çalıştırılabilir bileşenlere ve yazılım modüllerinin uygulamalarına dönüştürülerek yazılım sistemlerinin geliştirilmesi amaçlanmaktadır. Bu nedenle MDA, bir başka OMG (Object Management Group) standardı olan Meta-Nesne Tesisi (ing. Meta-Object Facility) (MOF) çerçevesine (OMG, 2003) dayanan model dönüşümlerini tanımlamaktadır.

MDA modellerine, geliştirme sürecinin model dönüşümü yapılmasından yazılım kodunun oluşturulmasına kadar her aşama entegre edilmiştir. Bu entegrasyonu sağlamak için MDA modellerin MOF tabanlı bir dilde ifade edilmesi gerekmektedir. Bu yükümlülük aynı zamanda modellerin MOF uyumlu depolarda saklanmasına, MOF uyumlu araçlarla ayrıştırılmasına, dönüştürülmesine ve gerekirse, Genişletilebilir İşaretleme Dili (ing. eXtensible Markup Language) (XML) Metaveri Değişimi (ing. XML Metadata Interchange) (XMI) formatına dönüştürülmesine ve bir ağ üzerinden iletilmesine izin vermektedir (OMG, 2003).

OMG, MDD ve önerilen MDA'yı desteklemek için bir dizi kılavuz sunmuştur (OMG, 2017). MDA'nın getirdikleri UML (tr. Birleşik Modelleme Dili, ing. Unified Modeling Language), MOF, XMI gibi çeşitli standartlarla ilgilidir. OMG MDA standardına dayanarak, metamodeller, modeller ve dönüşümler MDD yaklaşımlarının temel kavramlarıdır. Metamodeller, belirli bir alan için kavramlar arasındaki ilişkileri (örneğin, birleşme, bileşim, kalıtım vb.) tanımlar. Diğer yandan, modeller, metamodellerin meta elementlerini ve ilişkilerini içeren metamodel örnekleridir. Dönüşümler, başka bir metamodele dayanan bir modele (hedef model olarak adlandırılır) adapte olan başka bir modeli (kaynak model olarak adlandırılır) dönüştürmeyi mümkün kılar. OMG MDA tanımına göre, iki ana tip dönüşüm vardır, bunlar: modelden modele ve modelden koda dönüşümlerdir. Model dönüşümlerinde Platform Bağımsız Model (ing. Platform Independent Model)

(PIM) PIM seviyesine (yatay dönüşüm olarak adlandırılır) veya PIM'den Platform Spesifik Modele (ing. Platform Specific Model) (PSM) (dikey dönüşüm olarak adlandırılır) seviyesine dönüşümler yapılabilir. Modelden koda dönüşümde (veya modelden metne) bir PSM platforma özel bir koda dönüştürülür.

MDD, hedeflerini gerçekleştirmek için farklı yaklaşımlar kullanır. Bu yöntemlerden biri Alana-Özgü Diller'in geliştirilmesidir (ing. domain-specific languages) (DSL) (van Deursen et al., 2000; Mernik et al., 2005; Varanda Pereira et al., 2008; Fowler, 2011; Liu et al., 2012). DSL'ler, alanın gereksinimlerini karşılamak için belirli bir uygulama alanının kavramlarını ve terminolojilerini içeren dillerdir. Bir DSL, son kullanıcı programcılarına (alan uzmanları), alana özgü sorunlarla ilgili tanımlamalara izin verir. Başka bir deyişle, DSL'lerin son kullanıcıları gözlemlenen sorun alanları ile ilgili bilgi sahibidirler (Sprinkle et al., 2009) ve genellikle az programlama deneyimine sahiptirler.

MDD'yi gerçeklemeye yönelik bir diğer yaklaşım, modellerin görsel bir şekilde belirtilmesi ve tasarım kodları yerine ana eserleri temsil etmeleri nedeniyle, soyutlama seviyesini ve grafiksel sözdizimiyle tasarım kullanım kolaylığını artıran DSML'lerdir (Schmidt, 2006; Gray et al., 2007). DSML'ler, soyutlama, ifade edilebilirlik ve kullanım kolaylığını arttırmaktadırlar. DSML'nin ana fikri yazılım kodları yerine modellerin kullanılmasıdır ve kalsik DSL'lerden farklı olarak genellikle modeller görsel bir formatta belirtilmiştir (Schmidt, 2006; Gray et al, 2007). Çoğu DSML grafiksel gösterim kullanmaktadır, ancak sadece metin tabanlı olan DSML'ler de mevcuttur (Cuadrado and Molina, 2007; Mernik, 2013).

DSML'ler grafik bir sözdizim ile beraber, daha kolay tasarım ve değişiklik yapma gibi bazı avantajlar sunmaktadırlar. DSML geliştirme genellikle dil modeli tanımına dayanmaktadır (Strembeck and Zdun, 2009). Hedef etki alanını yansıtacak şekilde tanımlanması gereken alandan alınan kavramlar ve soyutlamalar bu kapsamda yapılmaktadır. Burada dil kavramları arasındaki ilişkilerin açıkça tanımlanması gerekmektedir. Böylece, modelleme dilinin soyut bir sözdizimi oluşturulur. Genel olarak, bir metamodel modeli tanımlar. Bir dil modelinin ek bölümleri, metamodel kullanılarak tanımlanamayan semantikleri tanımlayan kısıtlamalardan oluşur. Kısıtlamalar genellikle, Nesne Kısıtlama Dili (ing. Object Constraint Language) (OCL) gibi belirli bir dilde tanımlanır (OMG, 2017). Son kullanıcının modelleme ortamında, alan soyutlamalarının ve ilişkilerinin somut bir sözdizimi içinde sunulması ve bir modelleme bloğu olarak hizmet etmesi gerekmektedir. Eclipse GMF (The Eclipse Foundation, 2006) ve Sirius (The Sirius

Project, 2013) gibi özel yazılımlar kullanılırsa, bu modelleme ortamı otomatik olarak oluşturulabilir. Aksi takdirde, modelleme editörü manuel sağlanmalıdır. Daha sonra, alan çerçevesini oluşturmak için model dönüşümlerinin tanımlanması gerekmektedir. Alan çerçevesi, belirli bir ortamdaki DSML'lerin semantiklerini uygulamak için işlevler sağlayan bir platformdur. Genel olarak, semantikler çevirimsel semantikler (ing. translational semantics) ile verilir (Bryant et al., 2011).

Bir DSML'nin gelişimi genellikle dil modeli tanımlamasıyla gerçekleştirilir (Strembeck and Zdun, 2009). Yani, alandan gelen kavramlar ve soyutlamalar, hedef alanını (dil modeli) yansıtacak şekilde tanımlanmalıdır. Daha sonra dil kavramları arasındaki ilişkilerin tanımlanması gerekmektedir. Her ikisi de soyut bir modelleme dilini oluşturur. Genellikle bir dil modeli bir metamodel ile tanımlanır. Etki alanı soyutlamaları ve ilişkileri, grafiksel somut bir sözdizimine ihtiyaç duyar ve son kullanıcının modelleme ortamında (araçlar) bir modelleme bloğu olarak sunulur.

2.3 Araçlar ve Teknolojiler

MDD'de, modelleme ve dönüşümler için DSL'ler kullanılır. Modelleme, hedef alana uygun grafik ve/veya metinsel DSL'ler ile gerçekleştirilebilir. Yeni bir modelleme dili, alanın ihtiyaçlarına göre de tasarlanabilir. Modelden modele veya modelden metne dönüşümler, bu amaçla tasarlanmış DSL'ler tarafından gerçekleştirilebilir. Bu çalışmada, modelden koda dönüşüm geliştirmeleri DT alanında özgü olacak şekilde yapılmıştır. Eclipse Modelleme Projesi (ing. Eclipse Modeling Project), Eclipse topluluğu içinde model-güdümlü mühendisliği destekleyen çeşitli araçlara sahiptir (EMF, 2008). Bu projede EMF tabanlı bir araç olan Sirius (The Sirius Project, 2013) geliştirme platformu kullanılmıştır. Sirius modelleme ve kod oluşturma için platform sağlamaktadır. Ecore adı verilen bir metamodel ile modeller standart bir XMI formatında tasarlanabilir. Bu XMI format çıktısı, bir girdi olarak başka bir dönüşüm işleminde kullanılabilir.

Bu çalışmada DSML4DT modellerinden koda dönüşümleri sağlamak amacıyla Acceleo dili kullanılmıştır (Acceleo, 2018). Acceleo dili doğrudan Eclipse platformuna entegre olarak kullanılabilmektedir. Tüm Acceleo dönüşümleri Sirius ortamında yazılıp, doğrudan çalıştırılıp kod elde edilebilmektedir.

Tez kapsamında yapılan DT çalışmalarında geliştirme işlemleri için bir sürücü bilgisayarı gömülü sistem cihazı kullanılmıştır. Bu cihaz çift çekirdek

işlemcili, 1GB ana belleği ve 4GB sabit disk birimlerine sahiptir. Bu cihaz multimedya özelliklerini desteklemek için ses ve görüntü işlemlerini yapacak şekilde tasarlanmıştır. Toplu taşıma araçlarında kullanılan bu cihaz sürücülerin ücret toplama sistemlerini yönetmesine olanak sağlamaktadır (Kent Kart, 2018).

3. İLGİLİ ÇALIŞMALAR

Literatürdeki çalışmalar incelendiğinde, donanım sürücülerinin ve/veya kod üretiminin model-güdümlü gelişimi için bazı öneriler olduğu, ancak DT içeren mikroişlemci yazılımlarının oluşturulmasını dikkate almadıkları görülmektedir. (Chen et al., 2014)'deki çalışmada, Linux çekirdeği V2.6, sürücü ve "Makefile" dosyaları üretildiği ve tasarımda model-güdümlü tekniklerin kullanıldığı belirtilmiştir. Bununla birlikte, önerilen sistem DT ve DT yazılımlarını içermemektedir. Aynı araştırma ekibinin başka bir çalışmasında modele dayalı sürücü üretimi yapılmış, ancak DT yazılımları yine kullanılmamıştır (Chen et al., 2011). Farklı formatlarda elde edilen bilgi kaynaklarının bir multimedya kartı arayüzüne manuel olarak dönüştürülmesi ile sürücü kodu üretimleri bu çalışmada gerçekleştirilmiştir. Başka bir çalışmada (King et al., 2012), sürücü kodları "Bluespec Codesign" adlı dil kullanılarak üretilmiştir, ancak önerilen yöntem DT yapısını içermemektedir. Çalışma tek bir platform için MDSD desteğini sağlamaktadır. Bu tezde önerilen DSML4DT ile farklı mikroişlemci mimarileri için DT yazılımının geliştirilmesi mümkündür. Katayama et al.'ın (Katayama et al., 2000) çalışmasında, Unix benzeri sistemler için DT içermeyen sürücü kodları üretmektedir. Bu çalışmada, cihaz sürücülerinden soyutlanmış her sistem girişi yeniden tanımlanmış ve bir prototip sistem geliştirilmiştir. Bir uygulama örneği olarak, bir ağ cihazının kesme kontrolörünün üretimi verilmiştir. MDA tabanlı (Lecomte et al., 2011)'deki çalışmada, MOPCOM adlı bir yöntem kullanılarak gömülü sistemler için UML modelleri oluşturma kuralları açıklanmaktadır. Bu yöntemde soyut model, çalışma modeli ve ayrıntılı modelleme düzeyleri tanımlanmıştır. Modellemede MOPCOM kullanımı Çok-Giriş Çok-Çıkış (ing. Multiple-Input Multiple-Output) (MIMO) işlemidir ve bu çalışmaya kod üretimi dâhildir. Çalışmanın MDA yaklaşımı, DT veri yapısı ve ilişkili yazılım geliştirme sürecini desteklememekle birlikte, soyutlama düzeyleri ve uygulaması tezdeki çalışmaya benzemektedir.

DT yazılım geliştirme çalışmaları dikkate alındığında, (Likely and Boyer, 2008)'de, DT yazılımında düğüm tipleri, hiyerarşisi, sözdizimi konuları anlatılmış ve bir platformda DT'nin kullanımına ilişkin deneysel bir çalışma yapılmıştır. Buna ek olarak, bir sanal makine ve Çevresel Komponent Arabağlantı (ing. Peripheral Component Interconnect) (PCI) arayüzlerinin oluşturulmasında DT'nin kullanımını açıklayan çalışmalar (Nikkel 2016; Devigne et al., 2017) vardır. Ancak bu çalışmaların hiçbiri otomatik DT yazılımı üretimi yapmamaktadır. (Nicolescu and Mosterman, 2010) çalışmasında DT, Alan Programlanabilir Kapı Dizileri (ing.

Field Programmable Gate Arrays) (FPGA) tasarımında belirtilmiş olmasına rağmen, modelleme ve/veya otomatik üretim için bir yaklaşım yoktur. Bir DT derleyicisinin, "Altera SoC EDS" (RocketBoards, 2016) adlı bir üründe kullanıldığı görülmüştür. DT, FPGA entegreli bir sistem için burada üretilmektedir. Ancak, (RocketBoards, 2016)'da belirtildiği gibi, üretim yalnızca Linux'un tek bir çekirdek sürümü için mümkündür ve bu tezde hedeflenen farklı platformları destekleyen genel bir yapıya sahip değildir. (Jassi et al., 2016)'daki çalışmada, donanım sürücü kodu oluşturmak için IEEE 1685-2014 IPXACT (IEEE 1685-2014 standard, 2014) standardını kullanan gömülü sistemler için bir sistem önerilmiştir. Tek bir platform için video kameradan video oynatmanın örneğini bu çalışma içermektedir. Bu çalışmada EMF (Eclipse foundation, 2006) kullanılmış; DT üretiminin yapıldığı belirtilmiş, ancak yöntem, kapsam ve deneysel sonuç bölümlerinde DT üretimi ile ilgili bilgi elde edilememiştir. Önerilen yöntemde farklı platformlarda deneme yapılmamıştır. Ayrıca bu çalışmadaki DT desteği, tezde önerilen dil ve geliştirme süreci ile karşılaştırıldığında oldukça sınırlıdır. Bu tezde, (Jassi et al., 2016) çalışmasındaki DT tarafından desteklenmeyen Evrensel Seri Veriyolu (ing. Universal Serial Bus) (USB), Seri Çevresel Arayüz (ing. Serial Peripheral Interface) (SPI), Entegreler Arası Devre (ing. Inter-Integrated Circuit) (I2C) gibi birçok farklı arayüz için bileşenlerin modele dayalı geliştirilmesi dikkate alınmıştır. Son olarak, (Arslan vd., 2017) çalışmasında, gömülü bir sistem platformu için DT yazılımlarının nasıl geliştirilebileceği anlatılmıştır. Bu tezin çıkış noktasını da içeren bu çalışmada model-güdümlü DT geliştirme ihtiyacı vurgulanmıştır.

Yukarıda görüldüğü gibi model-güdümlü geliştirme ve/veya gömülü sistemlerdeki donanım sürücülerinin kodlarının oluşturulması için ilgili araştırma alanında çeşitli kayda değer çalışmalar bulunmaktadır. Ancak, bu çalışmalar DT tabanlı yazılımların gelişimini dikkate almamaktadır. Bu tez, farklı mikroişlemci mimarisine sahip sistemler için DT yapılarının ve ilgili yazılımların otomatik olarak üretilmesi için bir DSML'nin sağlanmasına yönelik ilk çabadır. Sunulan DSML, uluslararası "Devicetree" spesifikasyonlarına ve standartlarına göre DT tabanlı gömülü sistemlerin geliştirilmesini mümkün kılmaktadır.

4. DSML4DT SOYUT SÖZDİZİMİ

DSML'lerin soyut sözdizimi (ing. abstract syntax), bir alandaki kavramları ve bunların ilişkilerini tanımlar. Ayrıca, bir dilin soyut sözdizimi, dil tarafından sağlanan kavramların kelime dağarcığının model veya program oluşturmak için nasıl birleştirilebileceğini açıklar (Clark et al., 2004). MDD açısından soyut sözdizimler genellikle bir metamodel tanımı ile sağlanır. Metamodeller, modellerin neye benzemesi gerektiğini tanımlarlar. DSML4DT'in de soyut sözdizimi bir metamodel ile oluşturulmuştur. Bu bölümde ilgili ilgili metamodel anlatılmaktadır.

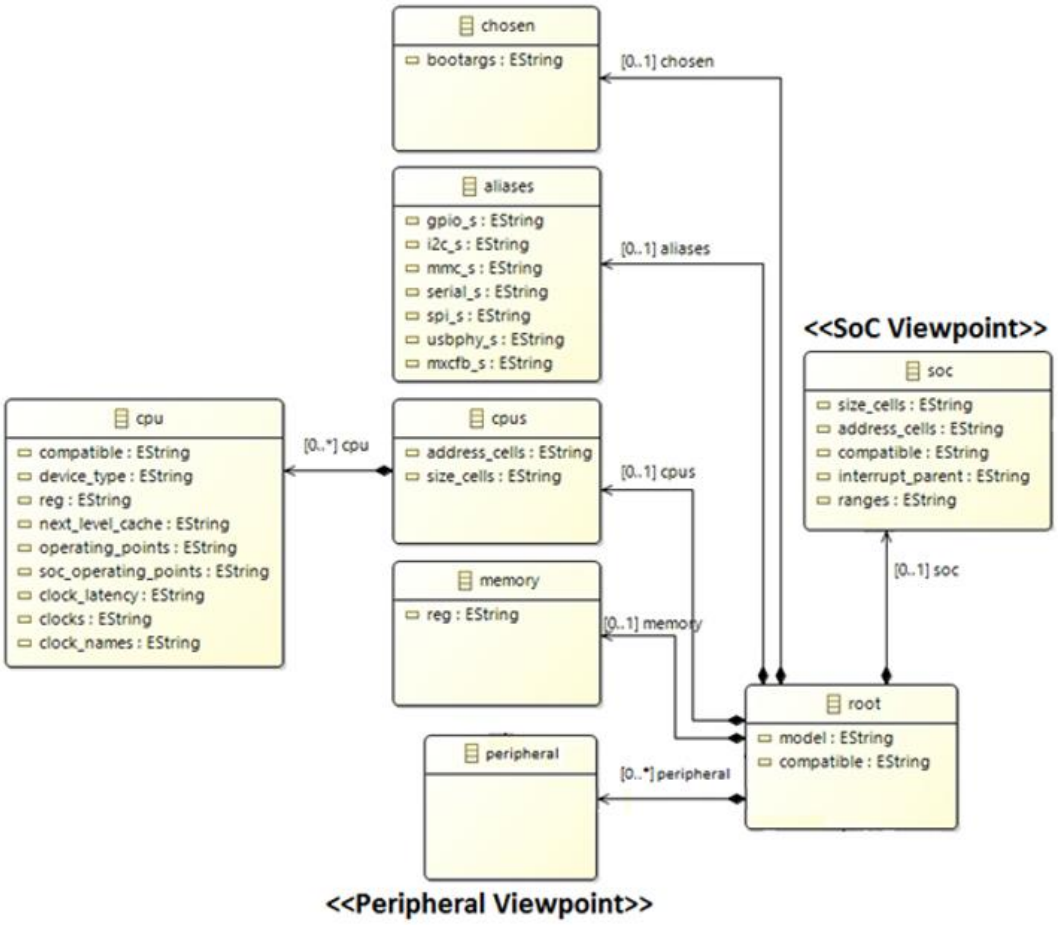
DSML4DT'nin soyut sözdizimini oluşturan metamodel (Devicetree Community, 2016)'da yer alan DT tanımlamaları göz önünde bulundurularak oluşturulmuştur ve kolay anlaşılması ve verimli kullanımı için beş farklı bakış açısına (ing. viewpoint) bölünmüştür. Bunlar Core, SoC, Aips_Bus, Spba_Bus and Peripheral olarak isimlendirilmişlerdir. Bu bakış açıları, DT standartlarındaki elemanların ilişkilerine göre bölünmüştür.

DT alanı elemanlarının farklı bakış açılarına göre gruplandırılması, DSML4DT'nin sözdiziminin geliştirilmesini kolaylaştırdığı gibi, bu sözdizimin DT geliştiricileri tarafından daha rahat yorumlanması ve daha etkin kullanılmasını da sağladığı söylenebilir. Örneğin, belirli DSML4DT bakış açılarının güncellenmesi (örneğin, elemanlar ve/veya ilişkilerin kaldırılması ve/veya değiştirilmesi) tek bir bakış açısı kullanımına ve büyük bir metamodel üzerinde çalışmaya kıyasla çok daha kolaydır. Dahası, bu çoklu görünüm yapısı ile mevcut sözdiziminin gelecekteki bazı özel ihtiyaçların olması durumunda genişletilmesi, geliştiriciler için de mümkün olmaktadır.

Tüm DSML4DT bakış açıları, aşağıdaki alt bölümlerde ayrıntılı olarak ele alınmıştır. Bir bakış açısının her diyagramında, üstmodel varlıkları ya da bir başka deyişle elemanlar sarı dikdörtgenlerle gösterilmiştir. "<<" ve ">>" karakterlerini kullanan elemanlar diğer bakış açıları ile olan ilişkileri göstermektedir. Başka bir deyişle, bu unsurlar bakış açıları arasında ortak unsurlardır ve onları birbirine uyarlar. Örneğin, Core bakış açısından, SoC meta-elemanı <<SoC Viewpoint>>'ten gelmektedir. Bu gösterim Şekil 4.1'den görülebilmektedir. Üstmodel elemanları orjinal tanımlarındaki İngilizce isimleri metinde geçmektedir. Normal metinden ayrılmaları için eğik olarak yazılmışlardır.

4.1 Core Bakış Açısı

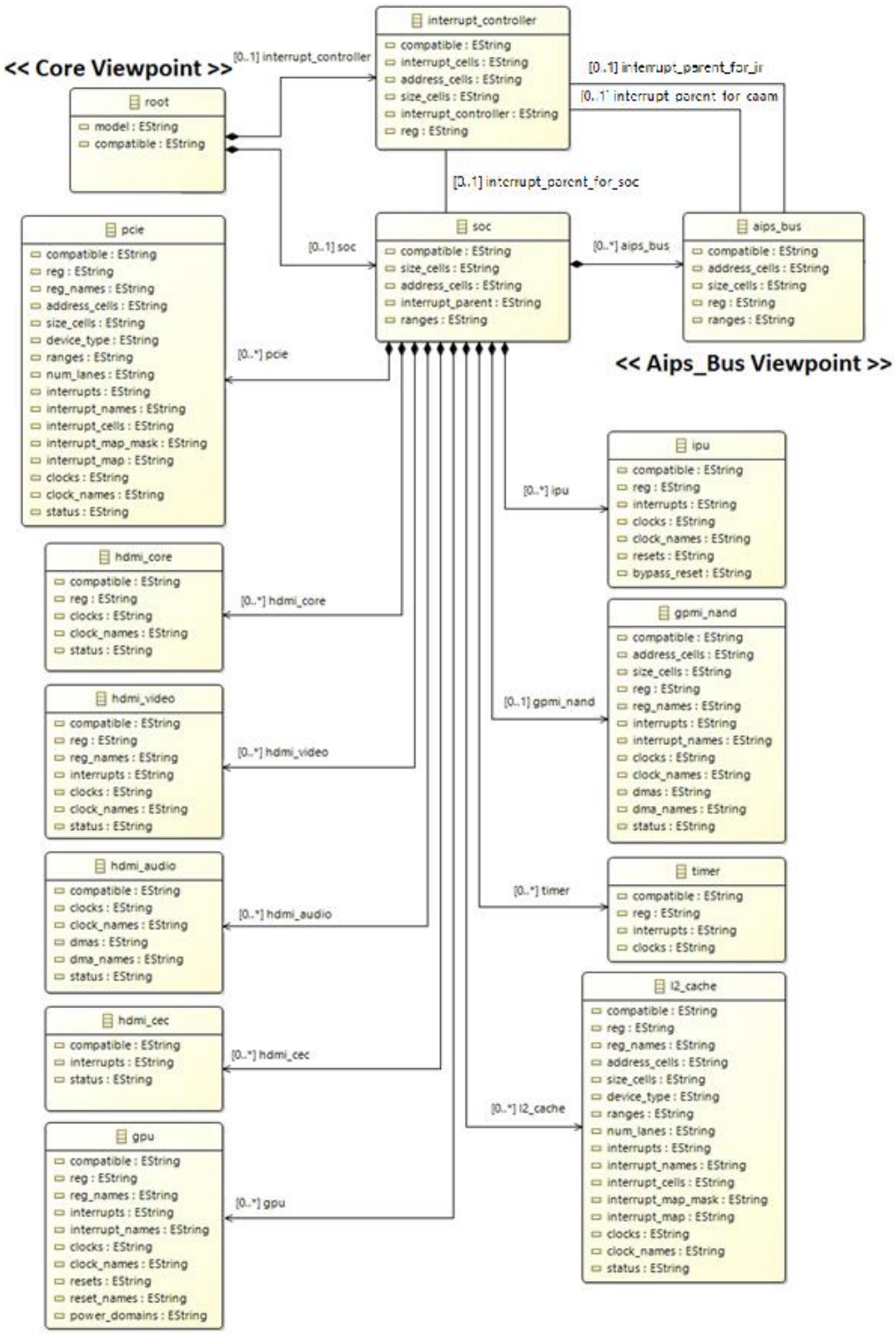
Core bakış açısının tüm öğeleri DT standartlarında tanımlanan düğümlerdir. Eleman *root*, tüm sistemin türetildiği temel düğümdür. *root* öğesi, kendinden türetilen diğer öğelerle sahiplik (ing. has-a relationship) ilişkisi içindedir. Gömülü sistemin işlemci ve belleği bu bakış açısında tanımlanmıştır. Çok çekirdekli işlemcilerde, her çekirdek birimi için gerekli türevler *cpus* öğesinden yapılır. Ek olarak, herhangi bir donanım ilişkisine sahip olmayan *aliases* ve *chosen* öğeleri burada bulunmaktadır. Bu öğeler, tüm DT yapısında kullanılacak kısaltmalar, tanımlamalar ve önyükleme parametre geçişlerini sağlarlar. Şekil 4.1, Core bakış açısı metamodelini göstermektedir.



Şekil 4.1 Core bakış açısı

4.2 SoC Bakış Açısı

Bu bakış açısı gömülü sistemlerde SoC entegre devrelerinin özelliklerine yönelik desteği içermektedir. Şekil 4.2'de, SoC bakış açısının yapısı görülebilmektedir. İlgili bakışaçısındaki tanımlamalar kullanılarak ses, görüntü ve zamanlayıcı gibi birçok SoC özelliklerinin parametre ayarları modellenebilir. Burada, *soc* ve *interrupt_controller* adlı iki temel öğe vardır. Bu öğeler, *root* elementi ile sahiplik ilişkisi içerisindedir. *aips_bus*, *ipu*, *gpmi_nand*, *timer*, *l2_cache*, *pcie*, *hdmi_core*, *hdmi_video*, *hdmi_audio*, *hdmi_cec* ve *gpu* öğeleri, *soc* elemanından türetilmiştir. *interrupt_controller* öğesi, gömülü sistemdeki tüm kesmelerin üretilmesini ve ayarlanmasını yönetir. Ayrıca, *soc* elemanında *interrupt_controller* öğesinden bir parametre kullanımı vardır. Bu iki öğe arasında *interrupt_parent_for_soc* ilişkisi mevcuttur. *soc* elementinin kesme ebeveyni *interrupt_controller* öğesidir. Ek olarak, farklı bir bakış açısı olan *aips_bus* elementi, *soc* elemanından türetilmiştir. *ipu* öğesi, görüntü işleme biriminin parametrelerini ayarlamak için DT yapısında bulunur. *gpmi_nand* öğesi, sabit disk için kullanılan Negatif-ve Kapısı (ing. Negative-AND) (NAND) belleğin parametrelerini belirlemek için kullanılır. *timer* elemanları SoC'deki zamanlayıcıları temsil etmektedir. Gömülü sistemin önbellek bilgisi için *l2_cache* elemanı türetilmiştir. Hızlı PCI (ing. PCI-Express) (PCI-E) arayüzünün konfigürasyon ayarlaması için, *pcie* elemanı DT yapısında bulunur. Tüm Yüksek Çözünürlüklü Multimedya Arayüzü (ing. High Definition Multimedia Interface) (HDMI) özellikleri için *hdmi_core*, *hdmi_video*, *hdmi_audio* ve *hdmi_cec* öğeleri bu bakış açısında bulunmaktadırlar.

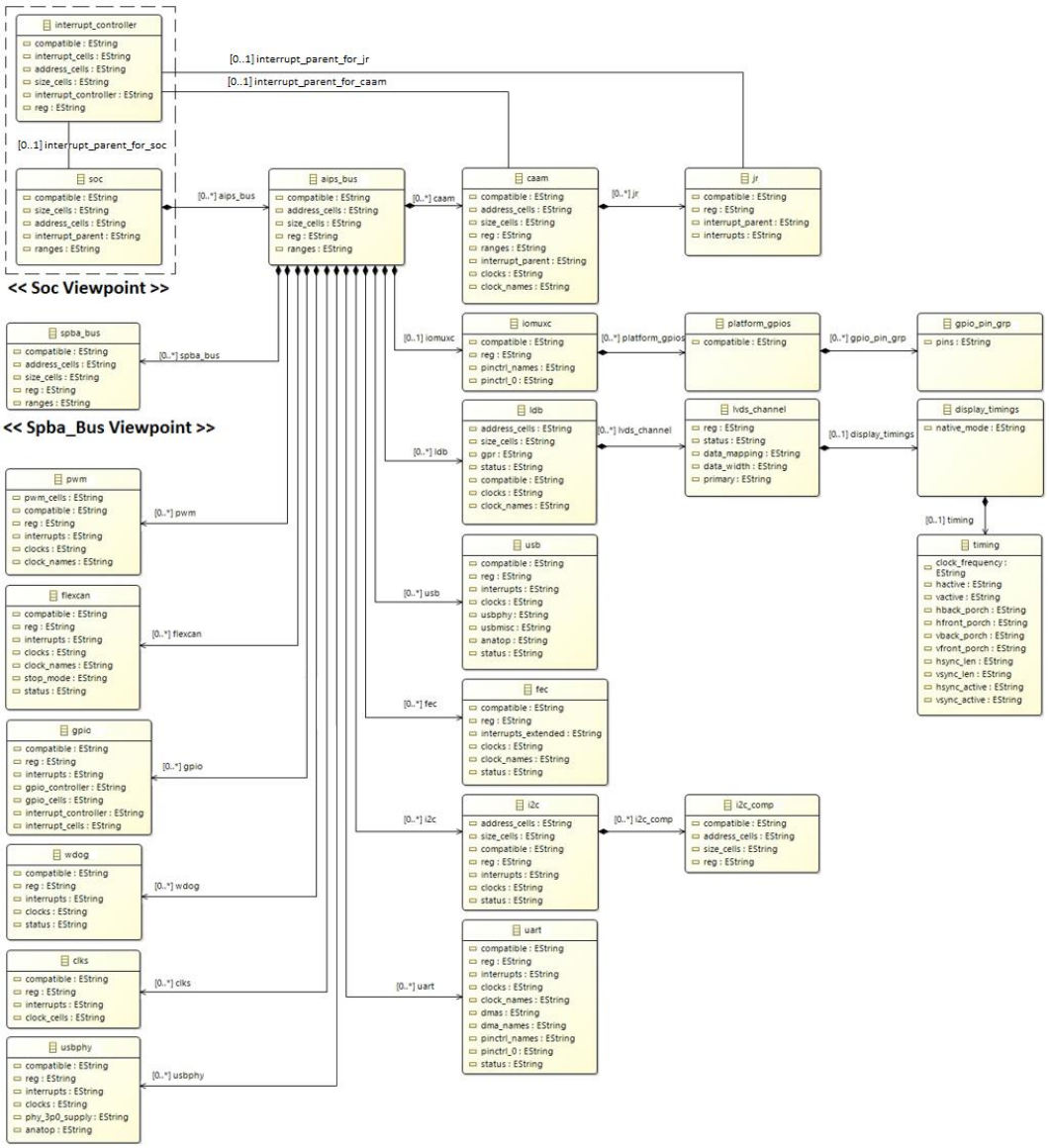


Şekil 4.2 SoC bakış açışı

4.3 Aips_Bus Bakış Açısı

Düşük bant genişliğine sahip SoC çevresel bileşenleri, gömülü sistemlerde Aips Bus arayüzü ile SoC birimleri ile iletişim kurmaktadırlar. Bu yapıyı

modellemeyi sağlayan Aips_Bus bakış açısı Şekil 4.3'te, görülebilmektedir. Aips_Bus bakış açısı ve öğesi, *soc* öğesinden türetilmiştir. Bu üstmodel elemanından birçok başka eleman üretilebilir. Gömülü bir sistemde *caam, iomuxc, ldb, usb, fec, i2c, uart, pwm, flexcan, gpio, wdog, clks, usbphy* ve *spba_bus* gibi elemanlar, SoC uyumluluğu ile *aips_bus* elemanından üretilebilirler. Bu özellikler sistemde mevcutsa, bu üstelemanların örnekleri hazırlanan sistem modelinde kullanılırlar. *caam* (tr. Kriptografik Hızlanma ve Güvence Modülü, ing. Cryptographic Acceleration and Assurance Module) (CAAM) ve *caam*'dan üretilen diğer elemanlar *interrupt_controller* öğesi ile *interrupt_parent* ilişkilerinde bulunur. *iomuxc* elemanı, SoC üzerindeki giriş ve çıkış pinlerinin özelliklerini çoğaltmak için kullanılır. *ldb* elemanı (tr. LVDS Görüntü Köprüsü, ing. LVDS Display Bridge) (LDB), LVDS (tr. Düşük Gerilimli Diferansiyel Sinyal, ing. Low Voltage Differential Signal) arayüzü ile Likit Kristal Ekranlarda (ing. Liquid Cyristal Display) (LCD) ayarlamaları yapmak için kullanılır. Bu üniteden farklı elemanlar üretilebilir. *usb* elemanı, sistemin USB denetleyici özelliklerini ayarlamak için kullanılabilir. Birden fazla USB hattı için çok sayıda öğe *aips_bus*'dan üretilebilir. *fec* elemanı, ethernet iletişim hattının özelliklerini girmek için kullanılır. *i2c* elemanı sistemdeki I2C arayüzünü kurmak için kullanılmaktadır. Sistemdeki her bağımsız I2C hattı için birden fazla eleman oluşturulabilir. *uarts* elemanı, Evrensel Asenkron Alıcı/Verici (ing. Universal Asynchronous Receiver/Transmitter) (UART) arayüzünün ayarlanması için tanımlanmıştır. Birden fazla satır için birden çok öğe *aips_bus* öğesinden türetilebilir. *pwm* elemanı, farklı çevresel birimlerin kontrolü için kullanılan Darbe-Genişlik Modülasyonu (ing. Pulse-Width Modulation) (PWM) sinyalinin kullanımı için yaratılmıştır. Gömülü sistem ağ yapısı olan ve Kullanıcı Alan Ağından (ing. Controller Area Network) (CAN) genişletilmiş Flexcan arayüzü için *flexcan* elemanı bulunmaktadır. Sistem tarafından birden fazla *flexcan* elemanı oluşturulabilir. Gömülü sistemler harici çevre kontrolü için Genel-Amaçlı Giriş/Çıkış (ing. General-Purpose Input/Output) (GPIO) bacaklarını kullanır. Bu bakış noktasında çoklu *gpio* elemanları oluşturulabilir. Gömülü sistemlerin kilitlenme durumunu kontrol eden donanım birimlerine watchdog denir. Bu birimlerin konfigürasyon ayarlamaları DT yapısında *wdog* elemanları ile yapılır. *clks* elemanı, SoC üzerindeki birimlerin saatlerini ayarlamak için tanımlanmıştır. USB arabiriminin fiziksel katmanını oluşturan *usb* elemanları bu bakış açısında mevcuttur.

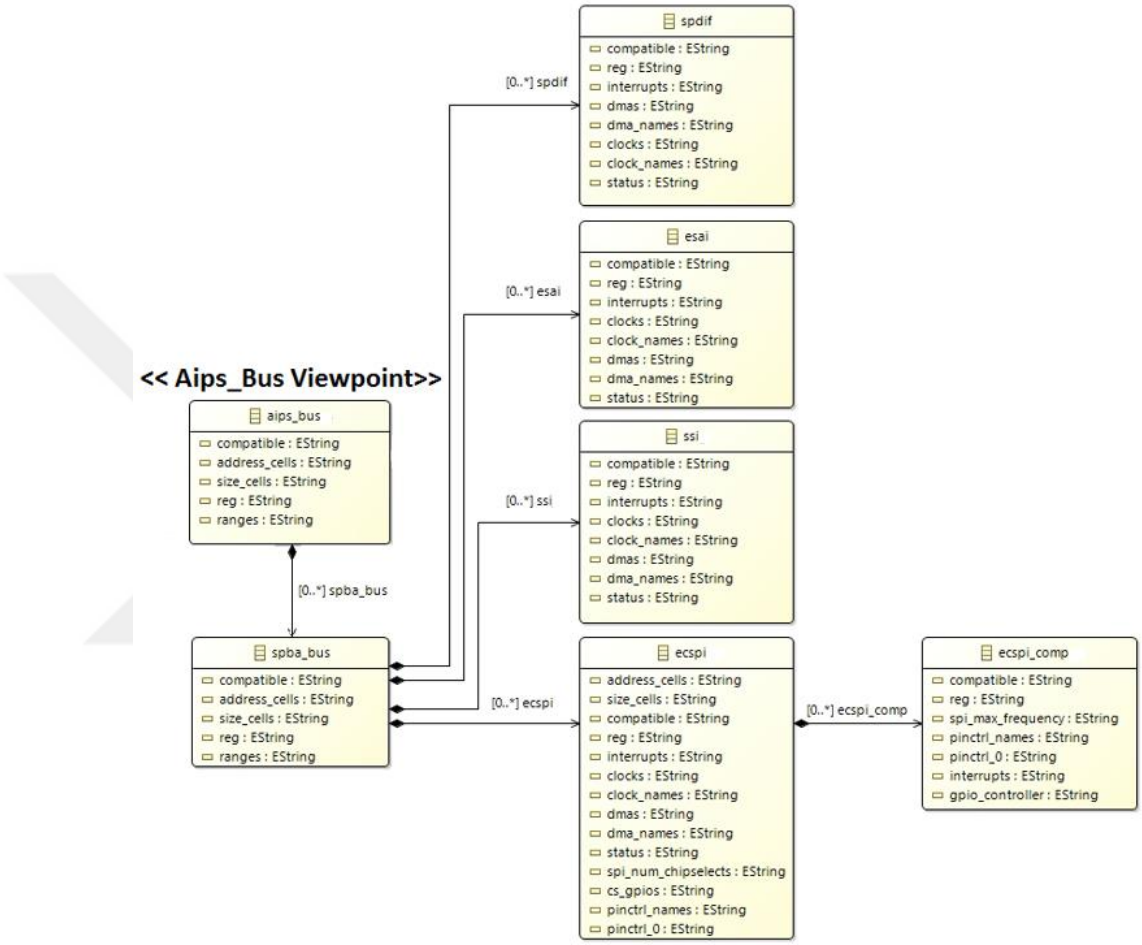


Şekil 4.3 Aips_Bus bakış açışı

4.4 Spba_Bus Bakış Açısı

Şekil 4.4'te, Spba_Bus bakış açısının yapısı görülebilir. Gömülü sistemlerde, paylaşılan bazı harici birimlerle iletişim kurmak için Paylaşımlı Çevresel Hat Arayüzü (ing. Shared Peripherals Bus Interface) (SPBA) veriyolu kullanılır. Bu arabirim Akıllı Doğrudan Bellek Erişimi (ing. Smart Direct Memory Access) (SDMA) çekirdeği ve çevre birimleri arasında iletişim kurar ve DT yapısı için birçok farklı alt birim kullanılmaktadır. Metamodelde bulunan *spba_bus*, *aips_bus* elemanından üretilir. *spdif, esai, ssi* ve *ecspi* gibi bazı birimler *spba_bus* elemanından üretilebilir. Bu elemanlar sistemin yapısına bağlı olarak birden fazla olabilirler. Sony/Phillips Sayısal Arayüzü (ing. Sony/Phillips Digital Interface) (SPDIF), bir dijital ses çıkışı, ayarlarının yapıldığı *spdif* elemanı *sbpa_bus*

elemanından oluşturulmaktadır. Pek çok ses özelliğini desteklemek için kurulan *esai* (tr. Geliştirilmiş Seri Ses Arabirimi, ing. Enhanced Serial Audio Interface) (ESAI) elemanı da bu bakış açısıyla kullanılmaktadır. Entegreler-Arası Ses (ing. Inter-IC Sound) (I2S) gibi ses sinyalleriyle desteklenen Seri Ses Arayüzü (ing. Serial Sound Interface) (SSI) ayarları, *ssi* elemanında yapılır. SPI sinyalini ayarlayan *ecspi* elemanı *spba_bus*'dan türetilmiştir.

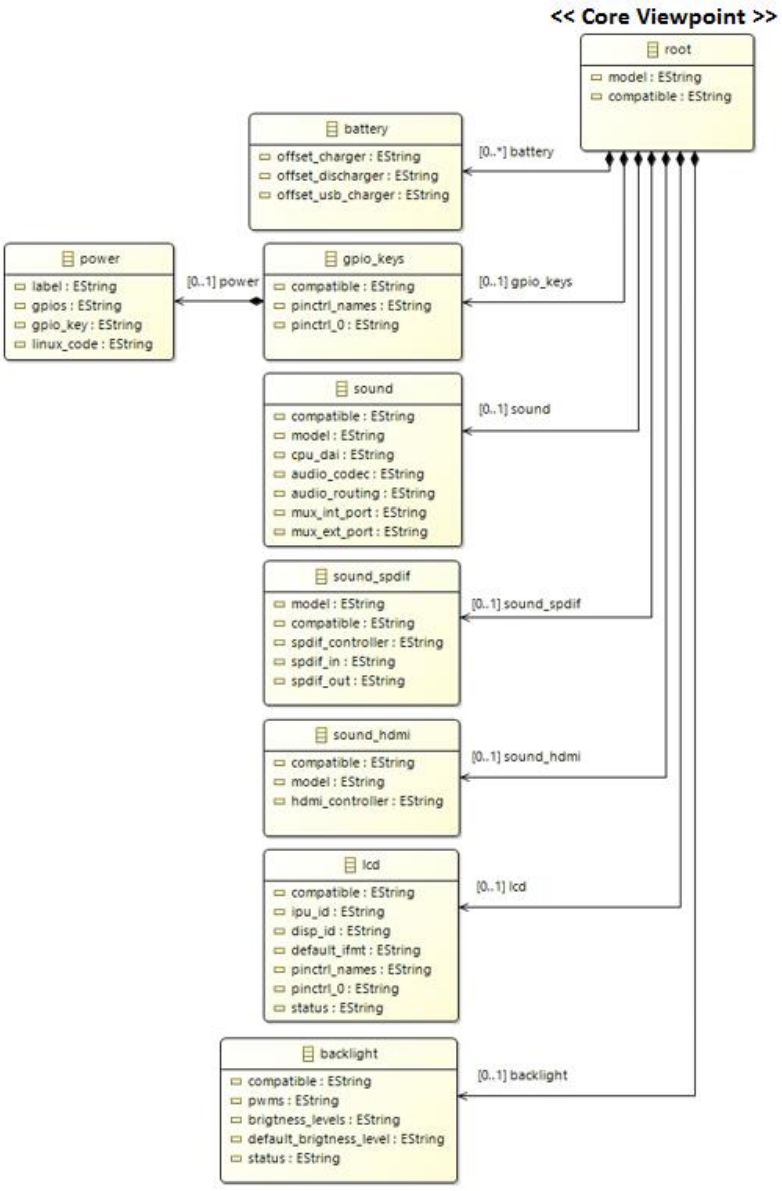


Şekil 4.4 Spba_Bus bakış açışı

4.5 Peripheral Bakış Açısı

SoC entegre devresinde olmayan ve SoC'a dışarıdan bağlı üniteler Peripheral (tr. çevresel) bakış açısında bulunur (Şekil 4.5). Bu bakış açısında, farklı ses ve video dönüştürücüleri gibi entegre devre çevre birimleri tanımlanmıştır. Bu elemanlar doğrudan *root* elemanından üretilir. Önceden açıklanan bakış açılarındaki elemanlar SoC entegre devresinde bulunmaktaydı. Bu bakış açısındaki birimler, benzer isimlere sahip olsalar bile, SoC'ta bulunmayan çevresel birimleri temsil eder. Bunlar *clocks, battery, gpio_keys, sound, sound_spdif, sound_hdmi, lcd*

ve *backlight* gibi öğelerdir. *clocks* elemanı, SoC entegre devresine bağlı saat sinyallerinin bilgilerinin girildiği bölümdür. Gömülü sistem herhangi bir bataryadan besleniyorsa, *battery* elemanı kullanılır. Sistemde açma/kapama düğmeleri gibi düğmeler varsa *gpio_keys* öğesi kullanılır. Her yeni özel düğmenin öğeleri *gpio_keys* öğesinden oluşturulur. Ses elemanları, sistemde bulunan herhangi bir ses entegre devresi için kullanılır. Sisteme harici olarak eklenebilen SPDIF ve HDMI dönüştürücüleri için öğeler vardır. Bu eleman isimleri *sound_spdif* ve *sound_hdmi* olarak belirtilmiştir. Sistemde bulunan Kırmızı Yeşil Mavi (ing. Red Green Blue) (RGB) LCD'ler için bir LCD öğesi de oluşturulmuştur. LCD'lerin arka ışığını ayarlamak için, *backlight* elemanı DT yapısında bulunmaktadır.



Şekil 4.5 Peripheral bakış açışı

5. DSML4DT SOMUT SÖZDİZİMİ

Soyut sözdizimi, dil üzerinde temsil edilen kavramları ve bu kavramlar arasındaki ilişkileri içerirken somut sözdizim (ing. concrete syntax) soyut sözdizimdeki kavramlar ve bunların örnek modeller üzerindeki temsilleri arasında bir haritalama sağlar. Somut sözdizim temel olarak dilin sunumunu ve inşasını kolaylaştıran bir dizi işarettir. Bu bölümde, DSML4DT'nin soyut sözdizim öğelerini grafiksel gösterimlere eşleyen grafiksel somut sözdizimi açıklanmaktadır.

Günümüzde, DSML'lerin somut sözdiziminin geliştirilmesi için görsel modelleme ortamları sağlayan çeşitli araçlar bulunmaktadır. Yerleşik genel kavramlara dayanan Grafik Modelleme Ortamı (ing. Graphical Modeling Environment) (GME) (GME, 2001) buna bir örnektir. Bu ortam genişletilebilir ve C++, Visual Basic, C# ve Python gibi genel amaçlı diller için kullanılabilir. Bu araçta hiyerarşi, çoklu yönler, kümeler, referanslar ve açık kısıtlamalar içeren büyük ölçekli ve karmaşık modeller oluşturmak için çeşitli kavramlar desteklenir. Kullanması kolaydır, ancak Eclipse GMF (Eclipse Foundation, 2006) veya Sirius'taki (The Sirius Project, 2013) gibi ayrıntılı modelleme yoktur.

Bir başka araç olan Freemind (Freemind, 2002), kullanıcı dostu bir arayüz sunmaktadır. Bu araç, test tasarımlarının projelerini ve görsel modellerini takip etme olanağı sunar. Bu araçta ayrıca yazı yazma ve beyin fırtınası becerileri de vardır. Ancak, bu araç modelleme için özel olarak tasarlanmamıştır.

MetaEdit+ (MetaCase, 1995) piyasada yaygın olarak bilinen bir araçtır. Geliştirme sırasında tek bir kullanıcı veya birden çok kullanıcı için metamodelleme ve modelleme ortamları sağlayan bir araçtır. MetaEdit+, geliştiriciler için kolay kullanım sağladığını iddia etmektedir. Ayrıca, bu araç MetaCase Corporation'ın desteği ile güçlü bir geçmişe sahiptir. Ancak, bu araç ne açık kaynaklı ne de ücretsizdir.

Microsoft firmasının, Visual Studio'nun bir parçası olarak yayımladığı bir araç piyasada mevcuttur (Microsoft, 2005). Bu araç sadece Windows işletim sisteminde çalışır. Somut sözdizimi sınırlıdır. Örneğin, sembollerin sadece bir geometrik şekli olabilir (Kelly and Tolvanen, 2008). Ancak, bu araç ayrıca açık kaynaklı veya ücretsiz değildir.

Başka bir araç olan Eclipse GMF (Eclipse Foundation, 2006), DSML'leri geliştirmek için ayrıntılı özellikler sunmaktadır. Çok çeşitli kullanışlı bileşenlere sahiptir. Önemli bir bileşen, geliştiricilerin meta-metamodel seviyesinde metamodeller tanımlamasına imkân veren Ecore bileşenidir. Diğer bileşenler, özellikleri ve somut sözdizimi elemanları için grafik özellikleri olan paletlerdir (ing. palette). Bu işlemler "Tooling Definition" ve "Graphical Definition" bileşenleri ile gerçekleştirilmektedir. Ek olarak, GMF haritalama bileşeni, üstvarlıklar ve grafiksel özellikler arasında haritalama sağlar. Platformlar, bu bileşenin araç kodunu oluşturarak uygulanabilir. Bu aracın dağıtımı ve kullanımı ücretsizdir.

Eclipse GMF aracı bu tezin temel gereksinimlerini sağlamaktadır. Ancak, bu tezdeki somut sözdizim üretme çalışmasında Eclipse EMF ve GMF teknolojilerine dayanan Sirius aracı (The Sirius Project, 2013) tercih edilmiştir. Sirius, Eclipse modelleme teknolojilerini kullanarak grafik modelleme ortamını kolayca oluşturmaya olanak tanıyan bir Eclipse projesidir. Sirius, Obeo ve Thales tarafından, modele dayalı mimari mühendisliği için özel ihtiyaçlara kolayca adapte edilebilen genel bir yapı oluşturmak üzere geliştirilmiştir. Sirius bakış açısı yaklaşımına dayanır. Böylece, karmaşık mimariler belirli alanlarda daha kolay geliştirilebilir. Sirius ile oluşturulan bir modelleme ortamı, kullanıcıların EMF modelleri oluşturmasına, düzenlemesine ve görüntülemesine izin veren Eclipse editörlerinden oluşur. Editörler, modelleme ortamının bütün yapısını, davranışlarını ve tüm baskı ve navigasyon araçlarını tanımlayan bir model ile tanımlanır. Bir Sirius modelleme ortamının bu tarifi, Eclipse IDE içinde dinamik olarak yorumlanır. Özelleştirme için özel gereksinimi desteklemek için, bu araç birçok yönden genişletilebilir. Bunlar, yeni türdeki sunumları, yeni sorgu dillerini, Eclipse veya başka bir sistemle etkileşime geçmek için Java kodunu arayabilmelerini sağlamaktadır.

Somut sözdizimi geliştirmek için tez çalışmalarında ilk olarak, soyut sözdizimi meta öğeleri için grafiksel temsiller ayarlanmıştır. Düğümleri Ecore dosyasındaki alan kavramlarına bağlamak için Sirius aracı kullanılmıştır. Çizelge 5.1, 5.2, 5.3, 5.4 ve 5.5'te DSML4DT'nin tanımlanan tüm bakış açılarının grafiksel gösterimleri verilmiştir. Tablolardaki sol sütunlarda soyut sözdizimindeki meta öğelerin adları bulunurken sağ sütunlarda DSML4DT sözdizimindeki semboller gösterilmiştir.

Çizelge 5.1 Core bakış açısı için somut sözdizim kavramları ve gösterimleri

Kavram	Gösterim
Root	
Chosen	
Aliases	
Cpus	
Cpu	
Memory	
Soc	

Çizelge 5.2 SoC bakış açısı için somut sözdizim kavramları ve gösterimleri

Kavram	Gösterim	Kavram	Gösterim
Soc		Gpu	
Root		Interrupt Controller	
Pcie		Aips Bus	
Hdmi Core		Ipu	
Hdmi Video		Gpmi Nand	
Hdmi Audio		Timer	
Hdmi Cec		L2 Cache	

Çizelge 5.3 Aips_Bus bakış açısı için somut sözdizim kavramları ve gösterimleri

Kavram	Gösterim	Kavram	Gösterim
Aips Bus		Ldb	
Soc		Usb	
Interrupt Controller		Fec	
Spba Bus		I2c	
Pwm		I2c Comp	
Flexcan		Uart	
Gpio		Jr	
Wdog		Platform Gpios	
Clks		Gpio Pin Grp	
Usb Phy		Lvds Channel	
Caam		Display Timings	
Iomuxc		Timing	

Çizelge 5.4 Spba_Bus bakış açısı için somut sözdizim kavramları ve gösterimleri

Kavram	Gösterim	Kavram	Gösterim
Spba Bus		Ssi	
Aips Bus		Ecspi	
Spdif		Ecspi Comp	
Esai			

Çizelge 5.5 Peripheral bakış açısı için somut sözdizim kavramları ve gösterimlerİ

Kavram	Gösterim	Kavram	Gösterim
Soc		Sound Spdif	
Battery		Sound Hdmi	
Gpio Keys		Lcd	
Power		Backlight	
Sound			

Ecore modelleri Sirius'ta yaratılmıştır, böylece görsel diyagramlarda kolayca eleman ve ilişkileri görülebilmektedir. Ecore modelleri geliştirme sürecinde kullanılmaktadır. Bu işlem sırasında, semboller hem paletler hem de şekiller için ayarlanır. Simge geometrisi açıklanmış ve bazı kısıtlama kontrolleri dikkate alınmıştır. Ortaya çıkan yapı, DT geliştiricilerinin DSML4DT somut sözdizimiyle eşleşen gerekli bakış açılarının her biri için modeller tasarlayabilecekleri bir grafik editörüdür.

DSML4DT sözdizimi araçları, kullanıcı modelleme işlemi sırasında bazı kısıtlamaları/denetimleri gerçekleştirmektedir. Bu kontrollerin bazıları metamodelden bazıları da görsel arayüz aracından gelmektedir. Bu kontroller sırasıyla "Model Kısıtlamaları" ve "Grafiksel Araçlar" olmak üzere iki bölüm halinde ele alınmıştır.

5.1 Model Kısıtlamaları

DSML4DT Ecore modellerinden gelen Sirius kısıtlamaları, tüm bakış açılarındaki örnek modeller için sağlanmıştır. Bu kısıtlamalar aşağıdaki gibi sınıflandırılabilir:

Bölme kısıtlamaları: Ecore'daki meta elemanlar arasındaki bileşimsel ilişki, başka bir elemanın üretildiği bir ilişkidir. Örneğin, Core bakış açısındaki *root* elemandan *memory* ve *battery* elemanları üretilebilir. Ancak, bu ilişkinin bulunamayacağı öğelerde bu gerçekleşmez.

İlişki sayısı kısıtlamaları: Örnek modeldeki elemanlar arasındaki ilişkilerin sayısı, Ecore'daki bire bir, bire çok, çoktan çoğa ilişkilere bağlı olarak kontrol edilir. Örneğin, *root* elemandan sadece bir *cpus* elemanı türetilebilir. Ayrıca, işlemci kısıtlama sayısı nedeniyle *cpu* öğelerinin sayısı daha yüksek olabilir.

İlişki kaynağı ve hedef kısıtı: İlişkinin yönü ilişkinin kaynağını ve hedefini tanımlar. Bu kısıtlama Ecore seviyesinde tanımlanmıştır. Örneğin, *root* ve *battery* arasındaki ilişki tersine çevrilirse, bu yapı modelde oluşturulamaz.

5.2 Grafiksel Araçlar

Metamodel kısıtlamalarına ek olarak, DSML4DT modellemesi, model oluştururken kullanıcıya tasarımda yardımcı olan bazı editöre bağlı kısıtlamalar içerir. Bu kısıtlamalar aşağıda listelenmiştir:

Bakış açıları arasında geçiş: Bu kısıtlama, farklı bakış açılarının editörleri arasında geçiş sağlayarak sistemin birliğini tesis etmektedir. Bu yapı sistemin adım adım oluşturulması sağlar. Araç, diyagram dosyaları oluştururken oldukça esnektir. Bir kullanıcı talebi durumunda, bakış açısı şemalarının her birini ayrı ayrı oluşturmak mümkündür. Örneğin, kullanıcı Core diyagramını tasarlamadan SoC diyagramı tasarlayabilir.

Birleşim: Bu özellik tüm öğeler için sistemde birleştirme sağlar. Editör diyagramları bakış açılarına dayanmaktadır. Öte yandan, sistem metamodeli bir bütün model olarak düşünülmelidir. Bu nedenle, tüm modelleme işlemi sırasında, benzersiz bir şekilde kullanılmak üzere, tanımlanan herhangi bir eleman bir listeye kaydedilmektedir. Bu, herhangi bir bakış açısı şemasına dahil edilebilecek tüm unsurları gösteren bir ağaç yapısına sahip olarak elde edilir. Örnek olarak, Core bakış açısı düzenleyicisinde oluşturulan bir *root* elemanı verilebilir. Bu *root* elemanı Peripheral bakış açısı editöründe kullanılmalıdır. Sirius aracı, bakış açısı yapılarında tanımlanmışsa, bu elementi Peripheral görünümüne de ekler. Öğeler, diyagramlarda gerektiğinde paletlerden sürükleyip bırakılarak kullanılabilir.

İlişki-element bütünlüğü: Aracın bu kısıtlamasına göre, herhangi bir örnek modelde oluşturulan bir öğe kaldırıldığında, tüm ilişkiler modelden kaldırılacaktır. Bu, tüm modelin bütünlüğünün korunmasını ve modelin bu değişikliklerden sonra tutarlı olmasını sağlamaktadır. İlişki sayısı, kaynak ve hedef ilişkileri ve ilişki-eleman kısıtlamalarının bütünlüğü temel kısıtlamalardır.

5.3 Doğrulama Kuralları

Sirius aracında, yukarıda açıklanan kısıtlamaların ve özelliklerin yanı sıra doğrulama kuralları (ing. validation rules) tanımlamak da mümkündür. Örneğin, bazı elemanların mutlaka diyagramlarda bulunması gerekiyorsa, bunlar için bir doğrulama eklenebilir. Çizelge 5.6, DT bakış açılarına göre hazırlanmış doğrulama kuralları listesini göstermektedir. "Validate Diagram" işlemi, DT görünüm noktalarına göre hazırlanmış olan DT şemalarında gerçekleştirildiğinde, hata mesajları bu kurallara göre görüntülenir.

Çizelge 5.6 DT yapısının bakış açıları için doğrulama kuralları

Kurallar	**Uyarı Bildirimleri**
Core_1	The chosen element must be in the diagram
Core_2	The aliases element must be in the diagram
Core_3	The cpus element must be in the diagram
Core_4	The memory element must be in the diagram
Core_5	The soc element must be in the diagram
Core_6	The root element should be named as “root”
Core_7	The chosen element should be named as “chosen”
Core_8	The aliases element should be named as “aliases”
Core_9	The cpus element should be named as “cpus”
Core_10	The memory element should be named as “memory”
Core_11	Least one cpu element must be in the diagram
Soc_1	The aips_bus element must be in the diagram
Soc_2	The interrupt_controller element must be in the diagram
Aips_bus_1	The spba_bus element must be in the diagram
Aips_bus_2	Least one jr element must be in the diagram
Aips_bus_3	Least one platform_gpios element must be in the diagram
Aips_bus_4	Least one gpio_pin_grp element must be in the diagram
Aips_bus_5	Least one lvds_channel element must be in the diagram
Aips_bus_6	One display_timings element must be in the diagram
Aips_bus_7	One timing element must be in the diagram
Aips_bus_8	Least one i2c_comp element must be in the diagram
Spba_bus_1	Least one ecspi_comp element must be in the diagram
Peripheral_1	Least one key element must be generated from the gpio_keys

6. KOD ÜRETİMİ İÇİN MODEL-KOD DÖNÜŞÜMÜ

DT sistemleri için çalıştırılabilir yazılım kodları oluşturmak amacıyla Model-Metin (ing. Model-to-Text) (M2T) dönüşümleri uygulanmıştır. Bu dönüşüm kuralları bir başka deyişle dilin işletimsel semantiğinin oluşturulmasına imkan vermektedir. DSML4DT modellerinin yorumlanmasını desteklemek için, bu çalışmada, Acceleo'da (Acceleo, 2018) M2T dönüşüm kuralları yazılmıştır. Acceleo, OMG MOF Model-metin Dönüşüm Dili (ing. Model to Text Language) (MTL) standardının pragmatik bir uygulamasıdır. Ayrıca, Acceleo dönüşümlerinin yazıldığı, ayrıştırıldığı, kontrol edildiği ve doğrudan Eclipse ortamından yürütülebildiği bir Eclipse eklentisi (Eclipse Foundation, 2006) aracını bu dil sağlamaktadır.

Acceleo, herhangi bir metamodele bağımlı olmadan kod oluşturma kurallarının tanımını ve bu kuralların yorumlanmasını sağlar. Bu sebepten dolayı, Acceleo kuralları DT kaynak kodlarının oluşturulması için çalışma zamanında platform spesifik DT modellerinde hazırlanmış ve uygulanmıştır. DT için oluşturulan kodlar gömülü işletim sistemi ortamında doğrudan gerçekleştirilebilir durumdadır. Aşağıda, bu çalışmada sağlanan M2T dönüşüm kurallarının bazı örneklerini verilmiştir.

Önceki bölümlerde açıklandığı gibi, DT yapısı *root* düğümü ile başlar. Bu düğüm DT yapısında bulunmalı ve özellikleri belirtilmelidir. M2T kodu da ilk önce *root* düğümü ile başlar. Liste 6.1'deki satır 1-3'te *root* düğümün oluşturulması gerçekleştirilir. Burada, *[aroot.model/]* ve *[aroot.compatible/]* Acceleo kodları doğrudan model yapısının bilgisini içe aktarır ve metne dönüştürür.

Core bakış açısındaki *root* düğümü gibi, *cpus* düğümü de DT yapısında bulunur. Modelde *cpus* ismini oluşturacak olan Acceleo kodu *[aroot.cpus.name/]* şeklinde yazılmıştır. Ayrıca, *cpus*'ta kesin olan adres hücreleri ve boyut hücreleri değerleri de oluşturulur. *cpu* çekirdek düğümleri, gömülü sistem platformuna bağlı olarak oluşturulur. Bu işlemci düğümleri, tek çekirdekli veya 2, 4 veya 8 gibi çok çekirdekli olabilir. Çok çekirdekli işlemcilerde, çekirdek sayısı kadar, işlemci düğümlerinin modelden üretilmesi gerekir. Liste 6.1'de, satır 9 gibi bir döngü oluşturulmaktadır. Böylece üretim çekirdek sayısına göre yapılabilir. Her döngünün sonunda, yineleyicinin değeri 1 arttırılır. Bu şekilde çekirdek sayısına göre çekirdek düğümleri üzerinde gezinmek mümkündür. Ayrıca, 10. satırda olduğu gibi, *if* kontrolü, modellenmiş *cpu* düğümlerinin boş olup olmadığını kontrol eder.

Dizinlenmiş *cpu* düğümü hakkında bilgi yoksa, kod üretimi yapılmaz. 11-27 satırlarında, *cpu* düğümlerinin öznitelikleri *i* indeks sayısına göre alınır ve M2T üretimi yapılır. Böylece, *cpus* altındaki *cpu* düğümleri ve gömülü platforma göre DT kodu üretimi yapılır.

Liste 6.1 *root, cpus* ve *cpu* düğümleri üreten Acceleo kodlarından bir alıntı

```
01    /{
02      model = [aroot.model/];
03      compatible = [aroot.compatible/];
04
05      [aroot.cpus.name/]{
06              address-cells = <[aroot.cpus.address_cells/]>;
07              size-cells = <[aroot.cpus.size_cells/]>;
08
09              [for (Sequence{1, 2, 3, 4, 5, 6, 7, 8})]
10              [if (aroot.cpus.cpu.name->at(i)->notEmpty())]
11              [aroot.cpus.cpu.name->at(i)/]{
12                      compatible = [aroot.cpus.cpu.compatible->at(i)/];
13                      device_type = [aroot.cpus.cpu.device_type->at(i)/];
14                      reg = <[aroot.cpus.cpu.reg->at(i)/]>;
15                      next-level-cache = <[aroot.cpus.cpu.next_level_cache->at(i)/]>;
16                      [if (aroot.cpus.cpu.operating_points->at(i)->notEmpty())]
17                      operating-points = <[aroot.cpus.cpu.operating_points->at(i)/]>;
18                      fsl,soc-operating-points = <[aroot.cpus.cpu.soc_operating_points->at(i)/]>;
19                      clock-latency = <[aroot.cpus.cpu.clock_latency->at(i)/]>;
20                      clocks = [aroot.cpus.cpu.clocks->at(i)/];
21                      clock-names = [aroot.cpus.cpu.clock_names->at(i)/];
22                      [/if]
23                      };
24
25              [/if]
26              [/for]
27      };
```

Liste 6.1, bir DT modelinin (bir DSML4DT örneğinin) çekirdek bakış açısındaki *root, cpus* ve *cpu* düğümlerine karşılık gelen DT kodlarının otomatik üretilmesi için yazılan Acceleo kod örneğini gösterir. DT sistemindeki tüm bakış açıları için Accecelo programı benzer şekilde geliştirilmiştir. Eclipse aracında, Acceleo programının bulunduğu dosyanın adı *generate.mtl*'dir. DT yapısının sözdizimi kuralları, tüm kod oluşturma aşamasında da uygulanır. Ek 1'de, 7. Bölüm'de tanıtılan sürücü bilgisayarı için kullanılan Acceleo programının tamamı görülebilmektedir.

7. DSML4DT TABANLI DT GELİŞTİRME YÖNTEMİ

Önceki bölümlerde tartışılan DSML4DT'nin özellikleri, DT geliştiricilerinin DT tabanlı gömülü sistemlerini görsel olarak tasarlayıp uygulayabilecekleri, modele dayalı bir DT geliştirme yöntemini oluşturmak üzere birleştirilmiştir. DSML4DT'ye dayanan, önerilen DT geliştirme yöntemi, tam DT uygulamaları için sistem modelleme ve kod oluşturma adımlarını içerir.

Sistem modelleme adımında, DT geliştiricileri DSML4DT sözdiziminin 5 farklı bakış açısını içeren sistemi çizmek için DSML4DT'nin tam işlevsel grafik düzenleyicilerini kullanırlar. Her bakış açısı, Bölüm 4'te açıklandığı gibi kendi paletine sahiptir. Bu adım, geleneksel yazılım geliştirme yöntemlerinde bir sistem tasarımını ima ediyor gibi görünse de, araç yalnızca sistem modelleme için bir Bilgisayar Destekli Tasarım (ing. Computer Aided Design) (CAD) sunmaz; aynı zamanda tasarımcıları daha doğru modeller sunmaya yönlendiren çeşitli kontroller de sağlar. Bu kontoller DSML4DT kısıtlamalarının somut sözdiziminden kaynaklanmaktadır. Bu adımın sonucu DT'nin geliştirilmiş platform bağımsız (DSML4DT) modelidir.

DT için yazılım kodu oluşturma, DSML4DT tabanlı geliştirme yönteminde yapılır. DT geliştiricileri kendi platformuna özgü DT yazılım modellerini, hedef DT dağıtım çerçevelerine/dillerine uygun olarak kullanırlar. Bu aşamada, modeller için kod üretimi yapılır. Bu şekilde, geliştiricilerin üst düzey başlangıç modelleri hedef dilde (DSML4DT'de) kodlara dönüştürülür. Acceleo'da (Bölüm 6'da ele alınan) yazılmış olan M2T dönüşüm kuralları, hedef DT modellerinde otomatik olarak yürütülür ve DT'nin uygulanması için kodlar elde edilir. DT geliştiricisinin hem kuralların içeriğini hem de yürütme ayrıntılarını bilmesi gerekmez. Geliştirici, yalnızca DSML4DT aracının kullanıcı arayüzü üzerinden kod oluşturma özelliğini seçer. Otomatik kod oluşturma adımının sonucu DT kaynak dosyaları olmaktadırlar. Bu dosyalar, ihtiyaç duyulan işletim sistemlerinde kullanılabilir.

Son olarak, bu DT geliştirme yöntemindeki adımların DSML4DT grafik aracı tarafından desteklendiğini belirtmek gerekir. DT geliştiricisi, sağlanan kullanıcı arayüzü ile istenen DT platformunu modelleyebilmektedir ve yukarıda tartışılan adımlara ait kalan tüm geliştirme aktiviteleri, aracın geliştirme ortamı içinde otomatik olarak gerçekleştirilebilmektedir. Bununla birlikte, herhangi bir aşamada, geliştirici, elde edilen eserleri detaylandırmak veya özelleştirmek isterse, geliştirme sürecine müdahale edebilmektedir.

Aşağıdaki alt bölümlerde, DSML4DT'ye dayanan bu MDD yönteminin kullanımını değerlendirmek ve sağlamak için bir çift çekirdekli gömülü sistem için bir DT'nin geliştirilmesi anlatılmıştır.

7.1 Örnek Çalışma: Bir Sürücü Bilgisayarı için DT Yazılımı

DT yazılımı geliştirmenin yapıldığı cihaz, toplu taşımada kullanılan bir sürücü bilgisayardır. Bu cihazda DT yapısının kullanılabileceği Android veya Linux işletim sistemleri bulunabilmektedir. Sürücü bilgisayarı RF (tr. Radyo Frekansı, ing. Radio Frequency) kartları okuma, ortam sıcaklığını izleme ve sesli anons yapma gibi işlemleri yapabilir. Kent Kart Ege Elektronik A.Ş. (Kent Kart, 2018) tarafından özel olarak tasarlanmış sürücü bilgisayar cihazı Şekil 7.1'den görülebilmektedir.

Şekil 7.1 Sürücü bilgisayarı cihazı

Sürücü bilgisayar cihazı, çift çekirdekli ARM Merkezi İşlem Birimi (ing. Central Processing Unit) (CPU) olan bir i.MX6 Dual Lite serisi (NXP Semiconductor, 2013) mikro işlemcisine sahiptir. Cihaz ayrıca 1GB Rasgele Erişimli Bellek (ing. Random Access Memory) (RAM) ve 4GB depolama ünitesine sahiptir. Multimedya sistemi desteği için SPI arayüzlü bellek, Güvenli Sayısal Hat (ing. Secure Digital) (SD) arabirimli Multimedia Kart (ing. Multimedia Card) (MMC), gigabit Ethernet, Gerçek Zamanlı Saat (ing. Real Time Clock) (RTC), USB arabirimi gibi bazı bloklar mevcuttur. Aygıtta bir güç yönetimi denetleyicisi

de kullanılır. Araçtaki kontak ve kilometre sayacı sinyalleri bu cihazda kullanılabilir. Sistemin sabit disk desteği, SATA arabirimi tarafından sağlanır. Sistem seri porttan açılabilir ve geliştirme çalışmaları UART hattından gerçekleştirilir. Araçlardaki CAN'e erişmek için, mikro işlemci CAN arayüzü kullanılır. Ayrıca bu cihazda kullanılabilecek kamera arayüzleri de vardır.

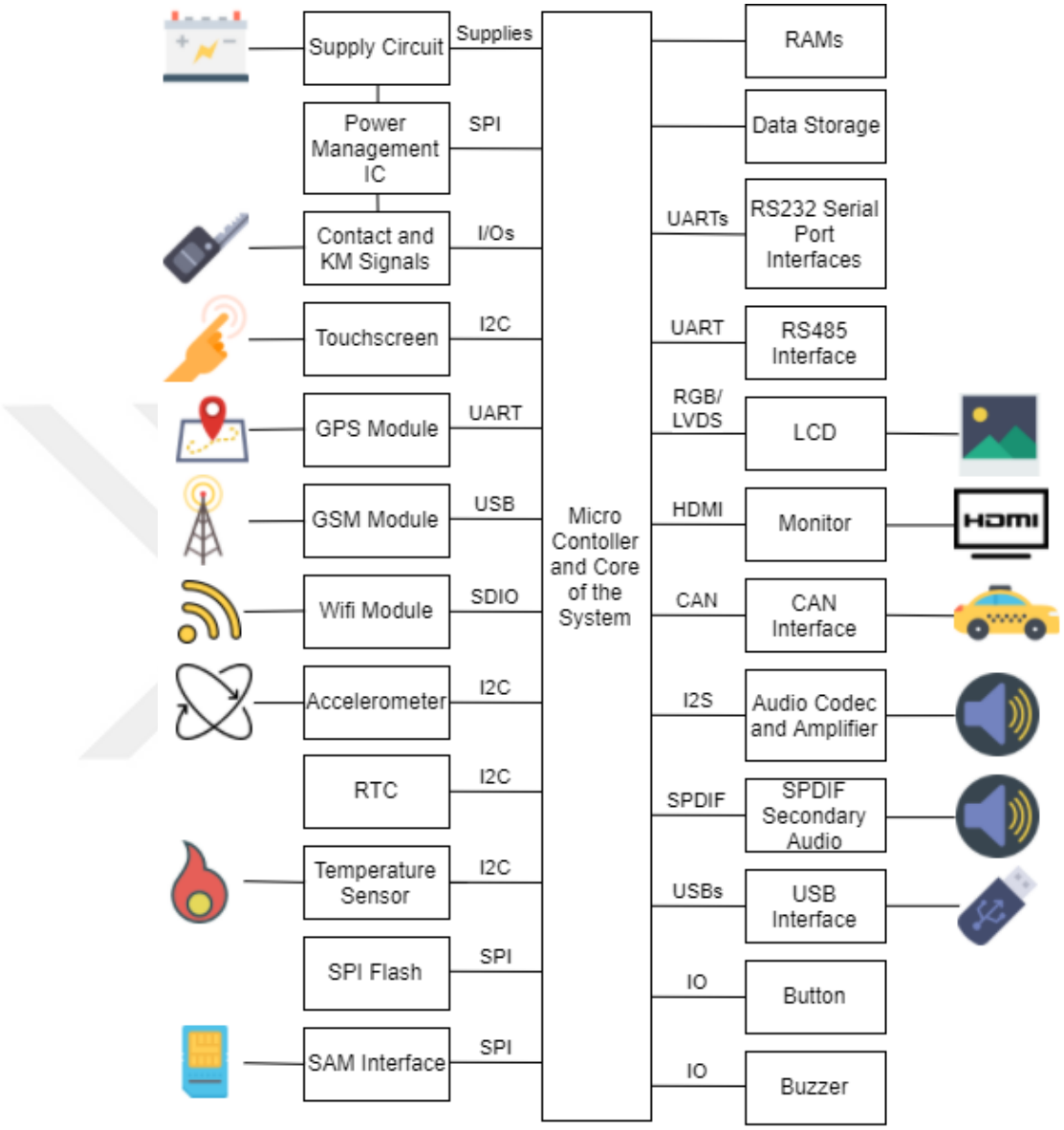
Sistemin bir multimedya cihazı olarak kullanılması için temel olarak ses, video ve iletişim arayüzleri bulunmaktadır. Ses çıkışı için I2S arayüzü ile çalışabilen bir ses dönüştürücü entegre devresi kullanılır. I2S'li mikro işlemciden gelen ses, ses dönüştürücüsünden geçer ve analog hat sinyali haline dönüşür. Hat sinyali ayrıca bir ses yükselticisine gelir ve yükseltilen sinyal hoparlörden duyulabilir. Harici bir mikrofon da bağlanabilir. Ana ses hattı ile kullanılabilecek bir Bluetooth ses sistemi de bu cihazda mevcuttur. Sistem ayrıca, ana ses çıkışından bağımsız olarak SPDIF arabirimi tarafından desteklenen ikinci bir ses çıkışına da sahiptir.

24 bit LCD'ler, RGB veya LVDS arayüzleri ile bu cihazda desteklenmektedir. LCD arka ışık kaynağı devresi bu cihazda bulunur. LCD arka ışığın parlaklığını ayarlamak için PWM sinyali de mikro işlemci tarafından sağlanır. Yüksek çözünürlüklü bir görüntü elde etmek için, cihaz HDMI arayüzü üzerinden monitör çalıştırabilmektedir. Dokunmatik ekran desteği için I2C hattı kullanılır. Dış dünya ile iletişim için RS232, RS485, USB ve gigabit Ethernet arayüzleri kullanılmaktadır. Cihazda toplam 5 adet seri iletişim hattı vardır.

Cihazda bir RF kart okuma/yazma devresi vardır. Güvenli RF okuma ve yazma için cihazda Güvenli Erişim Modülü (ing. Secure Access Module) (SAM) mevcuttur. İvme ölçer sensörü, Elektronik Olarak Silinebilir Sadece Okunur Bellek (ing. Electronically Erasable Programmable Read-Only Memory) (EEPROM), sıcaklık sensörü ve dijital potansiyometre de cihazda mevcuttur. Bu birimler I2C arayüzü tarafından kontrol edilir. Uyarılar için bir uyarı sinyali (buzzer) cihaz tarafından sağlanmaktadır. Cihazda bulunan bir buton ile dış dünyadan bilgi almak mümkündür.

Cihaz, bir Küresel Konumlandırma Sistemi (ing. Global Positioning System) (GPS) modülüne sahiptir. Böylece global konumlandırma bir UART arayüzü vasıtasıyla elde edilebilir. USB arayüzü üzerinden dış dünya ile iletişim sağlamak için bir Mobil İletişim İçin Küresel Sistem (ing. Global System for Mobile Communications) (GSM) modülü kullanılır. Wifi modülü, cihazda Güvenli Sayısal

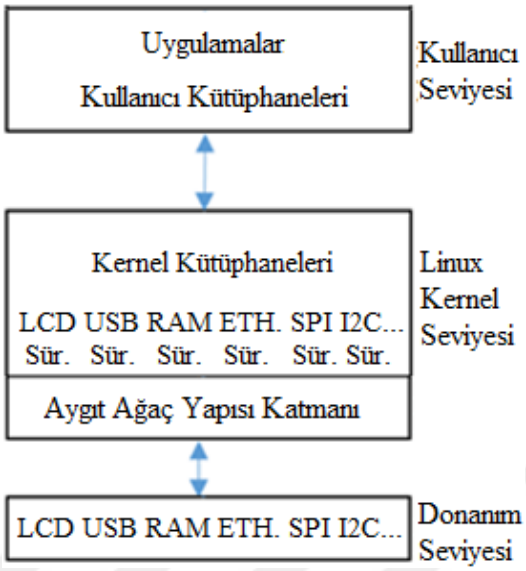
Giriş/Çıkış (ing. Secure Digital Input/Output) (SDIO) arayüzü üzerinden internet bağlantısı sağlamak için de kullanılır. Cihazın basitleştirilmiş genel donanım yapısı Şekil 7.2'den görülebilir.



Şekil 7.2 Sürücü bilgisayarı cihazının basitleştirilmiş diyagramı

Bu cihazda, tüm donanım birimleri, gömülü sistem mimarisinin aygıt ağacı katmanında açıklanmalıdır. Temel olarak aygıt ağacı yazılımı donanım ve işletim sisteminin sürücüleri arasında bulunmaktadır. DT kod dosyaları işletim sistemi çekirdeğinden bağımsız olarak derlenmektedirler. Tüm donanım parametreleri, önyükleme aşamasında, DT ikili kodundan işletim sistemi çekirdeklerine geçer. Çekirdeğin tüm sürücüleri ve kütüphaneleri bu donanım parametrelerini kullanarak

çalışır. Basitleştirilmiş Linux işletim sistemi katmanlı mimarisi Şekil 7.3'te görülebilir.

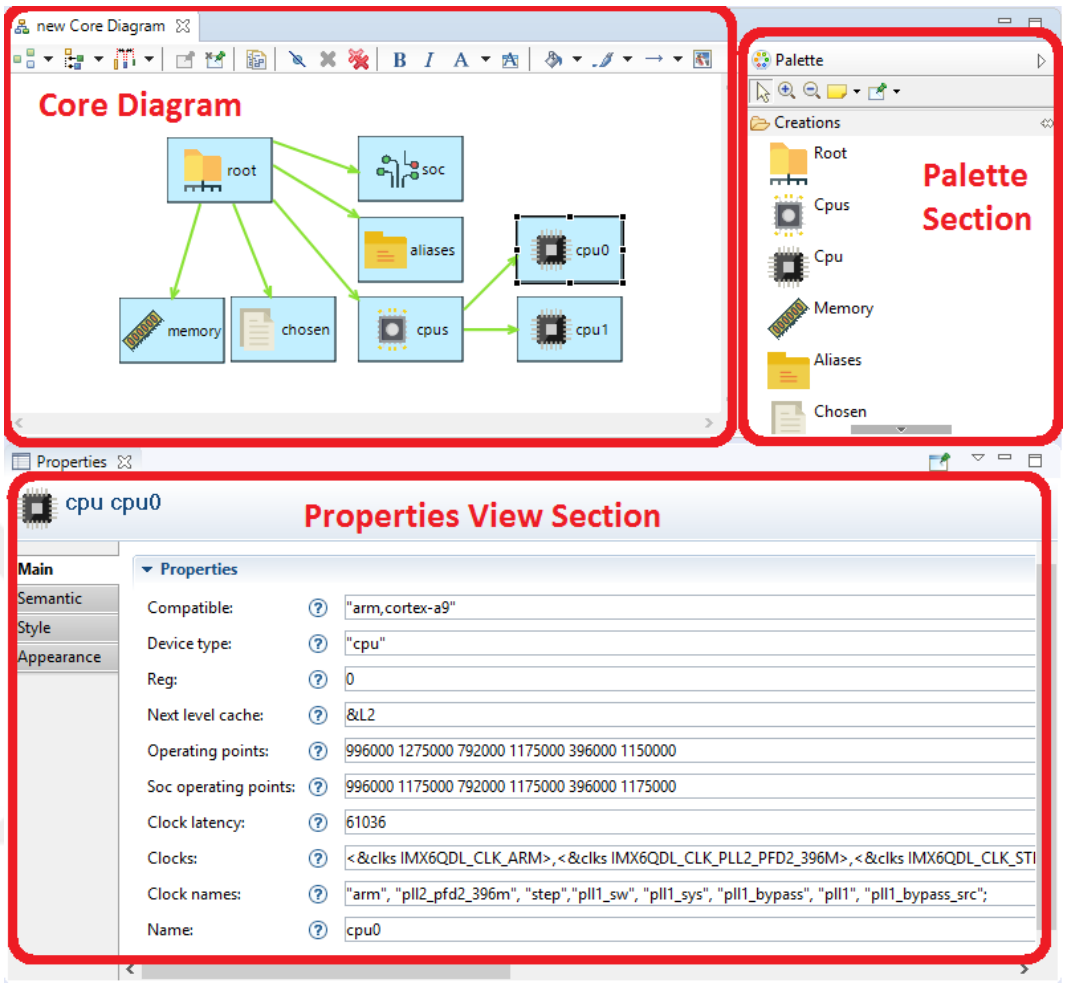


Şekil 7.3 Basitleştirilmiş işletim sistemi katmanlı mimarisi

7.2 Sürücü Bilgisayarı Cihazı İçin Sistem Modellemesi

DSML4DT model geliştirme aşamalarına detaylı olarak girmeden önce, geliştirme ortamına kısa bir giriş yapılması gerekmektedir. Daha önce de belirtildiği gibi DSML4DT'de toplam 5 bakış açısı vardır. Her bakış açısında model çizimi için diyagramlar oluşturmak mümkündür. Bu diyagramlarda model sembolleri, tanımlamalar ve isimler görülebilir. Tüm bakış açıları, model üretimi için Sirius Eclipse aracı desteği olan bir palet bölümüne sahiptir. Palet bölümünde, her bir bakış açısına özel düğümlerin simgeleri bulunur. Sürücü bilgisayarı için model geliştirme yapıldığında, palet bölümünde uygun sembollere sahip düğümler seçilir. Seçilen düğümler sürükle/bırak yöntemi ile eklenir. Bakış açıları, yalnızca kendisiyle ilişkili düğümler için simgeler içerir. Örneğin, Core bakış açısının çizim diyagramının palet bölümü sadece *root, cpus, cpu, memory, aliases, chosen* ve *soc* sembollerine sahiptir. Bazı ortak semboller farklı bakış açılarında bulunabilir. Örneğin, *soc* sembolü hem Core hem de Soc bakış açılarında bulunur. Bu nedenle, Core bakış açısı diyagramında oluşturulan *soc* düğümü, otomatik olarak Soc diyagramına da eklenir. Böylece modelin bakış açıları arasındaki tutarlılık sağlanır. Bu destek DSML4DT'de sağlanmıştır. Ek olarak, bakış açılarının model geliştirmesinde Özellikler (ing. Properties) bölümü vardır. Bu bölümde, sürücü bilgisayarının tüm donanım özellikleri yazılabilir. Böylece, tüm model özellikleri bu geliştirme aracıyla tamamlanabilir. DSML4DT'nin geliştirme aracı bölümleri

Şekil 7.4'te görülebilir. Burada Core bakış açısı diyagramı, Palette ve *cpu0* düğümü Properties görünüm bölümleri vardır.

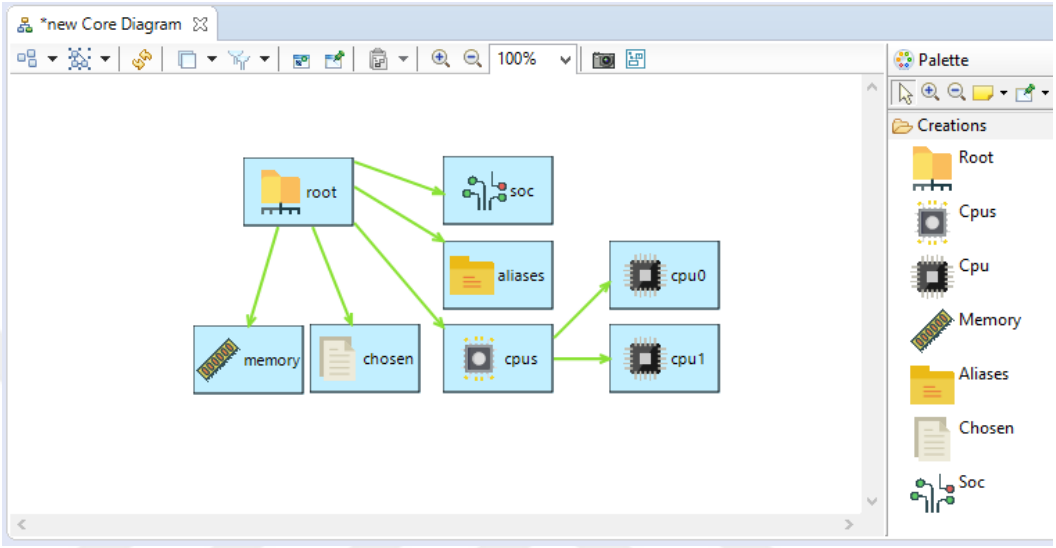


Şekil 7.4 DSML4DT için geliştirme ortamı

Durum çalışmasındaki sürücü bilgisayarı cihazına ait DT'nin DSML4DT tabanlı MDD yöntemine uygun olarak geliştirilmesine, farklı bakış açılarına göre sistem modelleri oluşturularak başlanmıştır. Bu örnekte, yukarıda açıklanan çift çekirdekli sürücü bilgisayarı uygulanmıştır. Bu vaka çalışması için, tüm bakış açıları kullanılmış ve sürücü bilgisayar cihazı için DT uygulaması tamamlanmıştır. Bu bakış açıları için modelleri temsil eden diyagramlar sırasıyla Şekil 7.5, 7.6, 7.7, 7.8 ve 7.9'da gösterilmiştir.

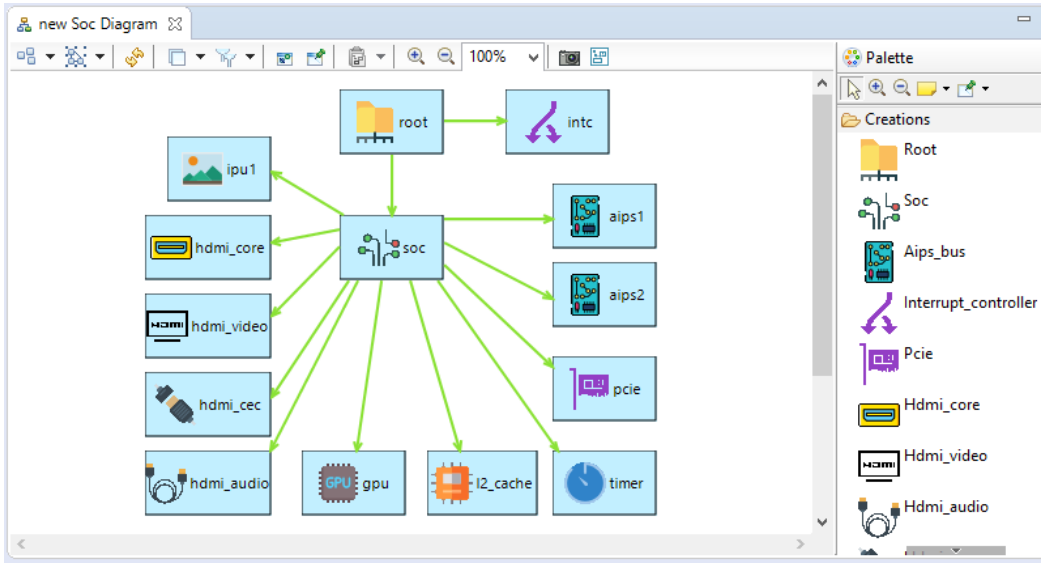
Yukarıda belirtilen sürücü bilgisayarına göre, DSML4DT'yi kullanarak gömülü cihazın DT'si modellenmiştir. Daha önce de belirtildiği gibi, DT yapısının başlangıcı Core bakış açısındadır. Bu nedenle, sistemin temel özelliklerini tanımlayan temel bakış açısı ilk olarak modellenmiştir. Burada, *root* düğümü oluşturulur ve DT yapısı için zorunlu olan *memory*, *chosen* ve *aliases* düğümleri

eklenmiştir. Sürücü bilgisayarında çift çekirdekli işlemci olduğu için *cpus* düğümünde 2 adet *cpu* düğüm oluşturulmuştur. Ayrıca, entegre özellikleri üzerine sistemin oluşturulacağı *soc* düğümü burada oluşturulmuştur. Tüm düğümlerin donanıma bağlı özellikleri bu aşamada girilir. DSML4DT grafiksel sözdizimi aracındaki sürücü bilgisayar cihazının Core bakış açısı için grafik modellemesi Şekil 7.5'te görülebilir.



Şekil 7.5 Core bakış açısı için DSML4DT grafik modelleme

Cihazın entegre özellikleri ile ilgili sistemin girildiği Soc bakış açısında, görüntü ve ses ile ilgili birçok düğüm eklenmiştir. *ipu* ve *gpu* düğümleri, sürücü bilgisayarı gömülü sistemindeki Görüntü İşleme Birimi (ing. Image Processing Unit) (IPU) ve Grafik İşleme Birimi (ing. Graphical Processing Unit) (GPU) birimleri için eklenmiştir. Tüm HDMI arabirim özelliklerinin tanımlandığı *hdmi_core*, *hdmi_video*, *hdmi_cec* ve *hdmi_audio* düğümleri bu bakış açısında yapılandırılmıştır. Cihazın önbellek ve zamanlayıcı birimleri için *l2_cache* ve *timer* düğümleri tanımlanmıştır. PCI-E sinyal arabirimi için *pcie* düğümü ve sistemdeki kesmeleri yönetmek için *intc* düğümü modele eklenmiştir. Sürücü bilgisayarı cihazında 2 adet Aips_Bus hattı bulunmaktadır. Bu nedenle *aips1* ve *aips2* oluşturulur. Aips_Bus hatları UART, SPI veya I2C gibi farklı gömülü sistem özelliklerini destekler. Oluşturulan *aips1* ve *aips2* düğümleri de Aips_Bus bakış açısının temel düğümleri haline gelmişlerdir. Tüm düğümlerin donanıma bağlı özellikleri, Soc bakış açısının modelleme aşamasında yapılandırılmıştır. DSML4DT grafiksel sözdizimi aracındaki sürücü bilgisayar aygıtının Soc bakış açısı için grafik modellemesi Şekil 7.6'dan görülebilir.



Şekil 7.6 Soc bakış açısı için DSML4DT grafik modelleme

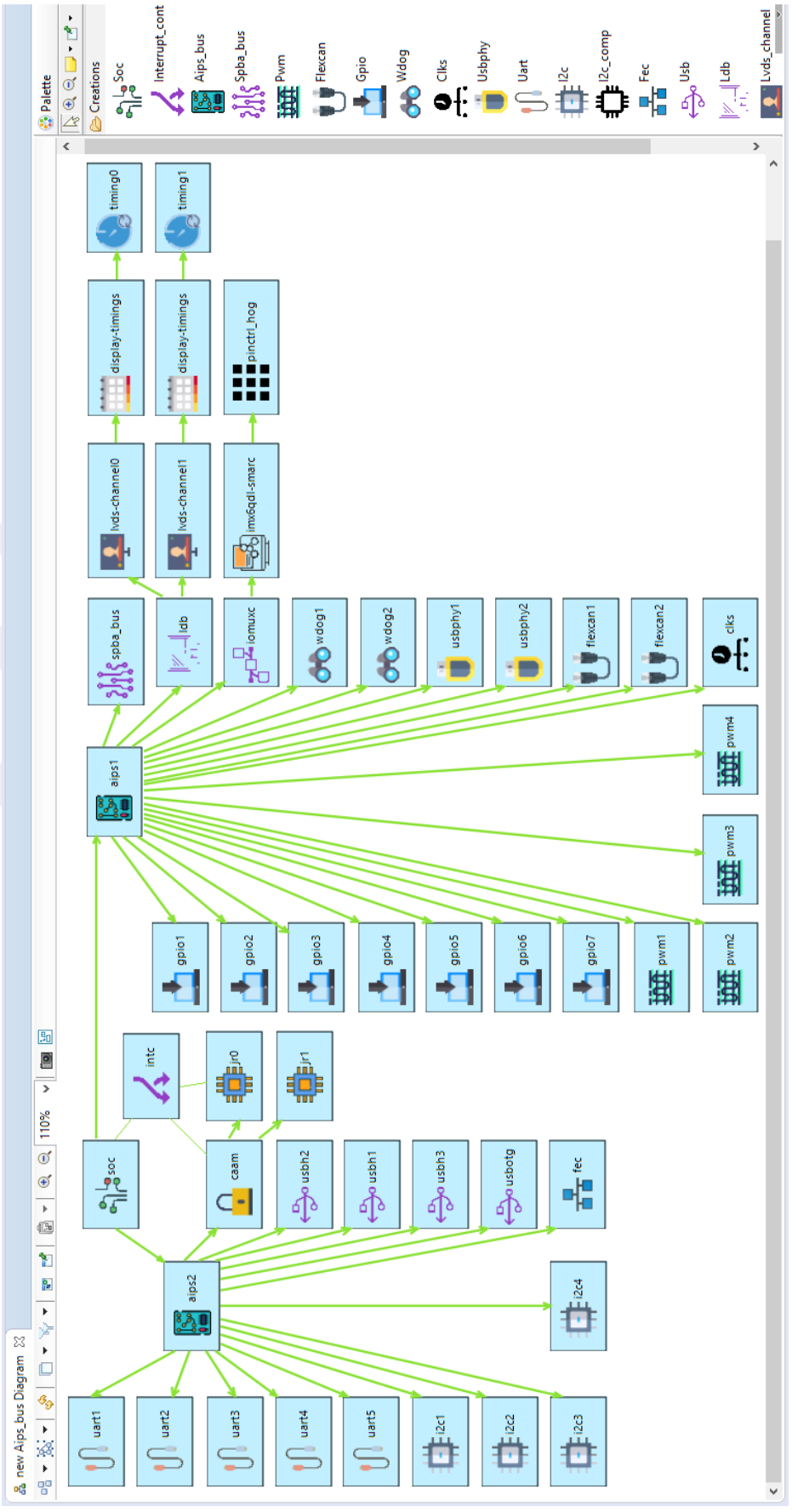
Soc düğümünden oluşturulan Aips_Bus bakış açısı, sürücü bilgisayarının modelindeki en büyük bakış açısıdır. Sürücü bilgisayar cihazının birçok özelliği ve farklı arayüzlerin desteği bu bakış açısıyla sağlanır. Cihazda 2 adet Aips_Bus hattı vardır. Bu hatlar, *aips1* ve *aips2* düğümleri ile modellenmiştir. *aips1* düğümünden 7 *gpio*, 4 *pwm*, 2 *flexcan*, 2 *usbphy* ve 2 *wdog* düğümleri üretilmiştir. Ayrıca *clks*, *iomuxc*, *ldb* ve *spba_bus* düğümleri *aips1*'den oluşturulmuştur. Spba_Bus bakış açısının başlangıç düğümü, bu bakış açısında oluşturulan *spba_bus* düğümüdür. *gpio1~7* düğümleri, sistemdeki tüm giriş ve çıkış bağlantı noktalarını tanımlar. Darbe genişlik sinyallemesinde kullanılan portlar *pwm0~4* düğümlerinde tanımlanır. *flexcan1* ve *flexcan2* düğümleri toplu taşıma araçlarında CAN'a bağlanmak için kullanılır. *wdog1* ve *wdog2* düğümleri, gömülü sistemin çökme durumları için cihazın sıfırlanmasını sağlarlar. Gömülü sistemdeki giriş ve çıkış portlarının özelliklerini çoğaltan *iomuxc* düğümü bu bakış açısındadır. *imx6dl-smarc* düğümünde, giriş/çıkış portları platform tarafından tanımlanabilecek şekilde oluşturulur. Bu düğümden, portların tanımlarının bulunduğu *pinctrl_hog* düğümü oluşturulur. *usbphy1* ve *usbphy2* düğümleri de USB hatlarının fiziksel arayüzünün tanımlanması için yaratılmıştır. LVDS sinyallerinin desteklenmesi için *ldb* düğümü bu bakış açısına eklenmiştir. Cihazda 2 LVDS kanalı desteği vardır, böylece *lvds-channel0* ve *lvds-channel1* düğümleri *ldb* düğümünden üretilir. Görüntü gösterim zamanlamalarını girmek için *display-timings* düğümünden *timing0* ve *timing1* düğümleri üretilmiştir.

aips2 düğümünden 5 *uart*, 4 *i2c*, 3 *usbh*, *fec*, *caam* ve *usbotg* düğümleri üretilir. Cihaz, UART sinyalleri üzerinden dış dünya ile iletişim kurabilir. Bu arabirimin tüm parametreleri, *uart1~5* düğümlerinde ayarlanır. USB Host ve USB

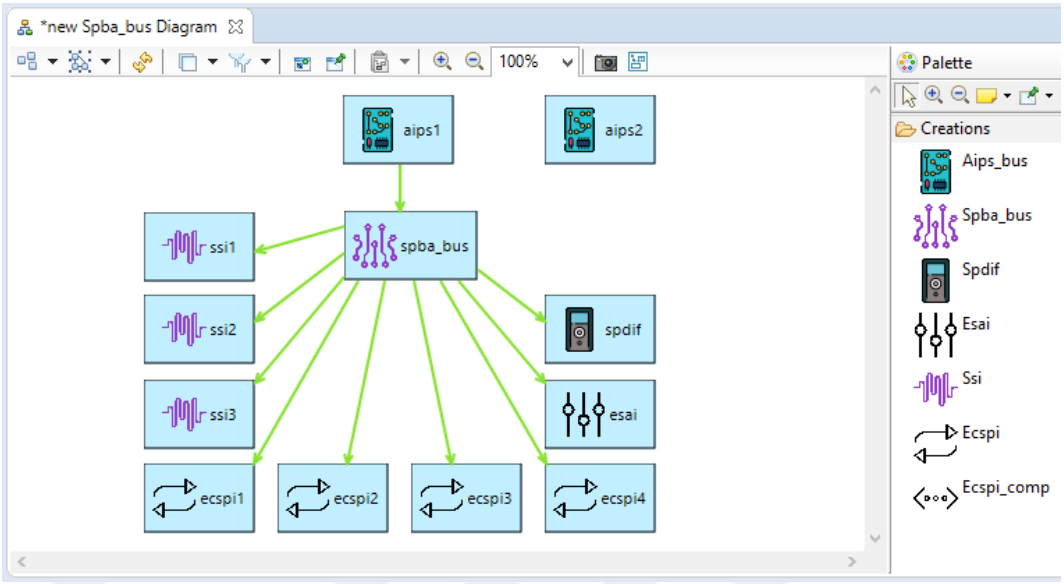
On-The-Go (USB OTG) iletişim arayüzlerinin ayarları *usbh1~3* ve *usbotg* düğümlerinde girilir. Tümleşik devre iletişim arabirimi olan I2C için *i2c1~4* düğümleri bu bakış açısında bulunur. Donanım düzeyinde güvenlik desteği için, *caam* düğümü oluşturulmuş ve *jr0* ve *jr1* düğümleri caam düğümünden türetilmiştir. Bu düğümler aynı zamanda *intc* düğümüyle de ilişkililerdir. Ethernet arayüzünün tanımı *fec* düğümünde yapılmaktadır. *aips2* düğümünden sistemin mimarisi nedeniyle, *spba_bus* düğümü üretilmez. *spba_bus* düğümü, *aips1* düğümünden üretilir. Tüm düğümlerin donanıma bağlı özellikleri Aips_Bus bakış açısında yapılandırılmıştır. DSML4DT grafiksel sözdizimi aracındaki sürücü bilgisayarı cihazının Aips_Bus bakış açısı için grafik modellemesi Şekil 7.7'de görülebilir.

spba_bus düğümü, Spba_Bus bakış açısı için *aips1* düğümünden üretilir. Bu bakış açısında 3 *ssi,* 4 *ecspi, esai* ve *spdif* düğümleri oluşturulur. Sürücü bilgisayarının bazı ses çıkış özellikleri Spba_Bus hattında bulunur. Bu nedenle, *ssi1~3, esai* ve *spdif* düğümleri Spba_Bus bakış açısında tanımlanmıştır. SPI iletişim hatları için parametre girişlerinin yapıldığı *ecspi1~4* düğümleri de vardır. DSML4DT grafiksel sözdizimi aracındaki sürücü bilgisayar aygıtının Spba_Bus bakış açısı için grafik modellemesi Şekil 7.8'de görülebilir.

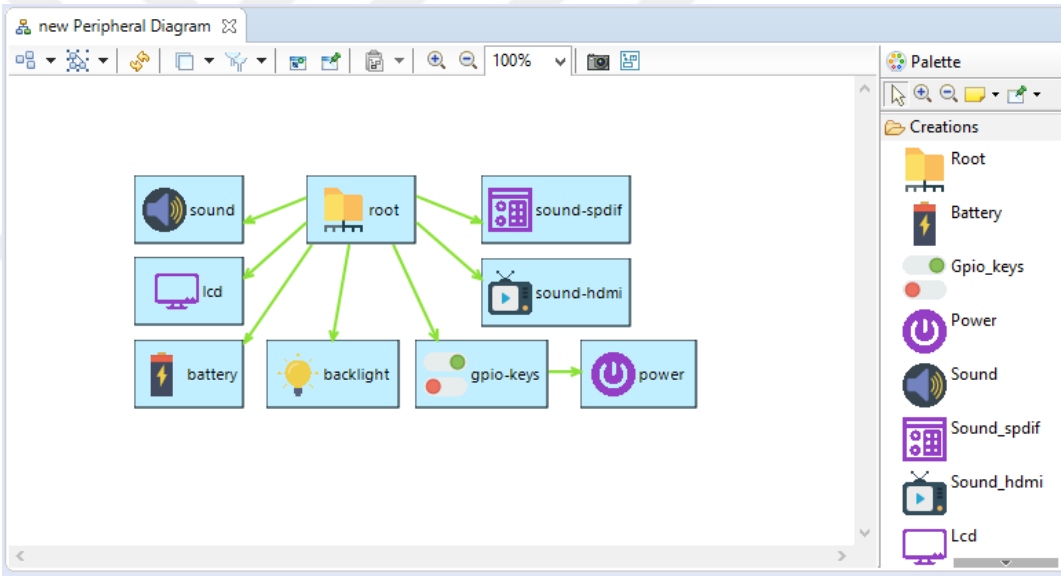
Son olarak, sürücü bilgisayarının Peripheral bakış açısı modeli yapılmıştır. Bu bakış açısı, SoC entegresi üzerinde sistem tarafından doğrudan desteklenmeyen özelliklere sahiptir. Burada harici entegre devreler tarafından desteklenen bazı özellikler bulunmaktadır. Ses dönüştürücü entegre devresini desteklemek için *sound* düğümü eklenmiştir. SPDIF dönüştücü entegre devresinin desteği için ses-*spdif* düğümü vardır. Harici RGB LCD'lerin arka ışık parlaklığı seviyesini ayarlamak için *lcd* düğümü ve arka ışık için *backlight* düğümü oluşturulmuştur. Pil desteği için *battery*, harici düğme desteği için *gpio-keys* ve HDMI sesi için *sound-hdmi* düğümleri eklenmiştir. *power* düğümü, bir Açma/Kapama düğmesi olarak kullanmak için *gpio-keys* düğümünden oluşturulmuştur. Buradaki tüm düğümlerin donanıma bağlı özellikleri Peripheral bakış açısında yapılandırılmıştır. DSML4DT grafiksel sözdizimi aracındaki sürücü bilgisayar cihazının Peripheral bakış açısı için grafik modellemesi Şekil 7.9'dan görülebilmektedir.



Şekil 7.7 Aips_Bus bakış açısı için DSML4DT grafik modelleme



Şekil 7.8 Spba_Bus bakış açısı için DSML4DT grafik modelleme

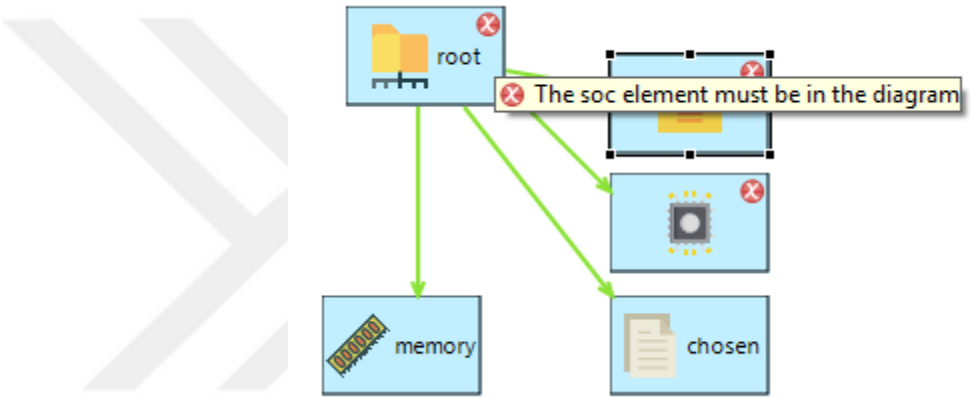


Şekil 7.9 Peripheral bakış açısı için DSML4DT grafik modelleme

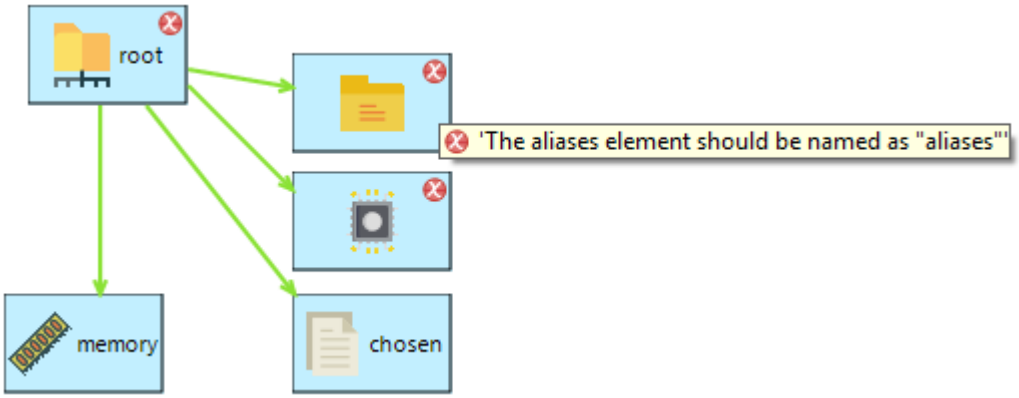
7.3 Modellenen Sistemin Doğrulanması

Model oluşturulduktan sonra doğrulama işlemi (ing. validation) yapılmalıdır. Bu süreç bazı kısıtlamaları kontrol etmeyi sağlamaktadır. Bölüm 5'te açıklanan ve Çizelge 5.6'da verilen doğrulama kuralları için kontrol prosedürü Sirius bazlı DSML4DT aracı ile yapılır. Araçta doğrulama işlemi için bir "Validate Diagram" seçimi vardır. Bu seçim yapıldıkta sonra eğer kural ihlali varsa hata mesajı alınır. Şekil 7.10, 7.11 ve 7.12 Core bakış açısı modelleme aşaması sırasında yapılan

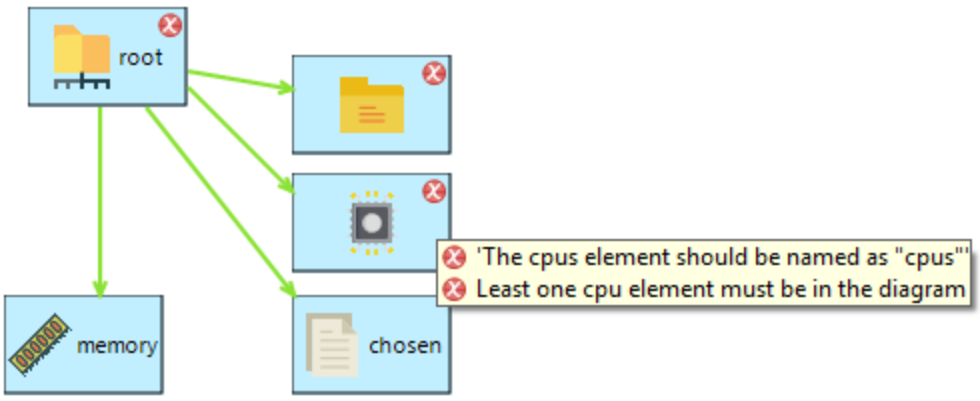
hataların bir kısmını göstermektedir. Core_5 kuralına göre, Core diyagramda mutlaka *soc* düğümünün olması gerektiği söylenmektedir. Diyagramda *soc* düğümü yoksa, Şekil 7.10'da görülebileceği gibi bir hata alınır. Core_8 kuralına göre, *aliases* düğümünün adı mutlaka "aliases" olmalıdır. Farklı bir isim girilirse hata mesajı alınır ve bu kural için alınan hata mesajı Şekil 7.11'de görülebilmektedir. Başka örnek vermek gerekirse, Core_9 kuralına göre, *cpus* düğümünün adı mutlaka "cpus" olarak girilmelidir. Core_11 kuralına göre ise, diyagramda en az 1 adet *cpu* düğümü bulunmalıdır. Core_9 ve Core_11 kurallarına göre kontrol hataları Şekil 7.12'de görülebilmektedir. Bir düğümde birden fazla hata mesajı alınabilir. Bu hataları düzelterek, tüm bakış açıları için kısıtlamalarla uygun model tasarımı yapılmış olmaktadır.



Şekil 7.10 Core_5 kuralı için doğrulama örneği



Şekil 7.11 Core_8 kuralı için doğrulama örneği



Şekil 7.12 Core_9 ve Core_11 kuralları için doğrulama örneği

7.4 DT Uygulaması İçin Kod Üretme

DSML4DT grafik aracı kullanılarak sürücü bilgisayarı cihazının modellenmesinden sonra ilgili DT yazılım kodları üretilmiştir. Daha önce Bölüm 6'da tartışılan M2T dönüşümleri bu amaçla kullanılmıştır.

Liste 6.1'deki Acceleo kodları, Core bakış açısında *root, cpus* ve *cpu* düğümleri için kod oluşturmayı içermektedir. DSML4DT aracı, sürücü bilgisayarının örnek hazırlanmış modelini girdi olarak alır. Bu araç, yazılı Acceleo programına göre kod üretir. Liste 6.1'de verilen Acceleo programına göre otomatik üretilen DT kodu Liste 7.1'de verilmiştir. 1-3 satırları kök düğümü için üretilen kodu göstermektedir. Bu satırlarda *model* ve *compatible* özellik bilgileri modelden elde edilir. 5-7 satırlarındaki bilgiler *cpus* düğümü modelinden elde edilir. 9-19 satırları *cpu0* düğümünden gelmektedir ve 21-26 satırlarındaki kod *cpu1* düğüm bilgisinden üretilmektedir. Oluşturulan kod, tüm bakış noktalarındaki tüm düğümlerden gelen bilgileri içerir. DT yazılımı, DT sözdizimi ve mimarisine uygun olarak üretilir. Sürücü bilgisayarı cihazı için toplam 906 satır kod oluşturulmuştur.

Ek 1'de bu sürücü bilgisayarı için kullanılan Acceleo programının tamamı görülebilmektedir. Ayrıca Ek 2'de sürücü bilgisayarı için üretilen kodun tamamı verilmiştir.

```
01    /{
02      model = Kentkart i.MX6 DualLite Smart Device Board;
03      compatible = "fsl,imx6dl-sabresd", "fsl,imx6dl";
04
05      cpus{
06              address-cells = <1>;
07              size-cells = <0>;
08
09              cpu0{
10                      compatible = "arm,cortex-a9";
11                      device_type = "cpu";
12                      reg = <0>;
13                      next-level-cache = <&L2>;
14                      operating-points = <996000 1275000 792000 1175000 396000 1150000>;
15                      fsl,soc-operating-points = <996000 1175000 792000 1175000 396000
                                                    1175000>;
16                      clock-latency = <61036>;
17                      clocks = <&clks IMX6QDL_CLK_ARM>,
        <&clks IMX6QDL_CLK_PLL2_PFD2_396M>,
        <&clks IMX6QDL_CLK_STEP>,<&clks IMX6QDL_CLK_PLL1_SW>,
        <&clks IMX6QDL_CLK_PLL1_SYS>,<&clks IMX6QDL_PLL1_BYPASS>,
        <&clks IMX6QDL_CLK_PLL1>,<&clks IMX6QDL_PLL1_BYPASS_SRC>;
18                      clock-names = "arm", "pll2_pfd2_396m", "step","pll1_sw", "pll1_sys",
                                      "pll1_bypass", "pll1", "pll1_bypass_src";
19                      };
20
21              cpu1{
22                      compatible = "arm,cortex-a9";
23                      device_type = "cpu";
24                      reg = <1>;
25                      next-level-cache = <&L2>;
26                      };
```

Liste 7.1 Core bakış açısından üretilen kod örneği

.

8. DEĞERLENDİRME

Bu çalışmada, Kent Kart Ege Elektronik A.Ş. Ar-Ge Merkezi'nde çalışan DT geliştiricilerinin DSML4DT'yi yazılım geliştirmede kullanımlarını göz önüne alan bir deneysel değerlendirme gerçekleştirilmiştir. Değerlendirme çalışmalarının, klasik kod merkezli geliştirme yerine DSML4DT kullanmanın kod üretimine olan katkısının açıklığa kavuşturmaya yardımcı olabileceği düşünülmektedir. Bu amaçla, her ne kadar çok-etmenli sistemlere ait DSML'lerin değerlendirilmesi için önerilmiş olsa da dil yapılarının ve DSML'lerin çeşitli boyut ve ölçütlere göre sistematik olarak değerlendirilmesinde de kullanılabilecek (Challenger et al., 2016)'deki DSML değerlendirme yöntemi bu tezdeki değerlendirme için uyarlanmıştır.

8.1 Değerlendirme Yaklaşımına Genel Bakış

(Challenger et al., 2016)'deki değerlendirme boyutları ve kriterleri göz önünde bulundurulduğunda, bu tezdeki değerlendirmenin kapsamı, Geliştirme Alt Boyutunu (ing. Development Sub Dimension) kapsamaktadır. Bu nedenle, bu boyut için değerlendirme Geliştirme Süre Performansı ve Kod üretim performansı DSML4DT için ölçülmüştür.

Bu çalışmada 5 adet Kent Kart Ege Elektronik A.Ş. DT geliştiricisi tezde önerilen dili ve buna bağlı MDD yöntemini değerlendirmiştir. Değerlendiriciler, gömülü sistem yazılım geliştirme uzmanlarıdır. Tüm değerlendiriciler, yazılım modelleme ve DT geliştirme gibi konularda bilgi sahibidir. Değerlendirme gönüllü katılan bu değerlendiriciler vaka çalışmasında öncelikle geliştirilmesi planlanan DT yazılımını manuel (doğrudan baştan sona kodlayarak) geliştirmiştir. Bu süre boyunca genel yazılım geliştirme araçlarını ve teknolojilerini kullanmışlardır. Daha sonra aynı değerlendiriciler aynı DT yazılımını bu sefer DSML4DT'yi ve buna ait modelleme aracını kullanarak geliştirmişlerdir. Değerlendirme sonuçları bu vaka çalışmasından elde edilmiş ve analiz edilmiştir. Değerlendiricileri çalışma hakkında bilgilerindiren, çalışmalara gönüllü katılımları için hazırlanan ve her bir değerlendiricinin imzaladığı "Bilgilendirilmiş Onam Formu" Ek-3'te verilmiştir.

Vaka çalışmasının yürütülmesi düşünüldüğünde, önce tüm değerlendiriciler bir örnek dahil olmak üzere DT için bir inceleme yapmıştır. Bu adım, DT alanlarına aşina olma düzeylerinin değerlendiriciler için mümkün olduğunca eşit olmasını sağlamaktadır ve değerlendiricilere ilişkin geçerlilik tehdidinin ortadan kalkmasına

yardımcı olmaktadır. Bundan sonra, tüm değerlendiriciler DSML4DT ve aracını kullanmak üzere kısa bir eğitimden geçmişlerdir. Ardından, tüm değerlendiriciler için vaka çalışmaları ve gereksinimleri üzerine bir sunum yapılmıştır. Son olarak, değerlendiriciler, kullanım durumunu analiz etme, sistemi tasarlama/modelleme, kod oluşturma ve test etme adımlarını ile DT'yi geliştirmiştir. Her geliştirici ve her adım için geçen süreler kaydedilmiştir. Sonuçlar aşağıdaki bölümde analiz edilmiştir.

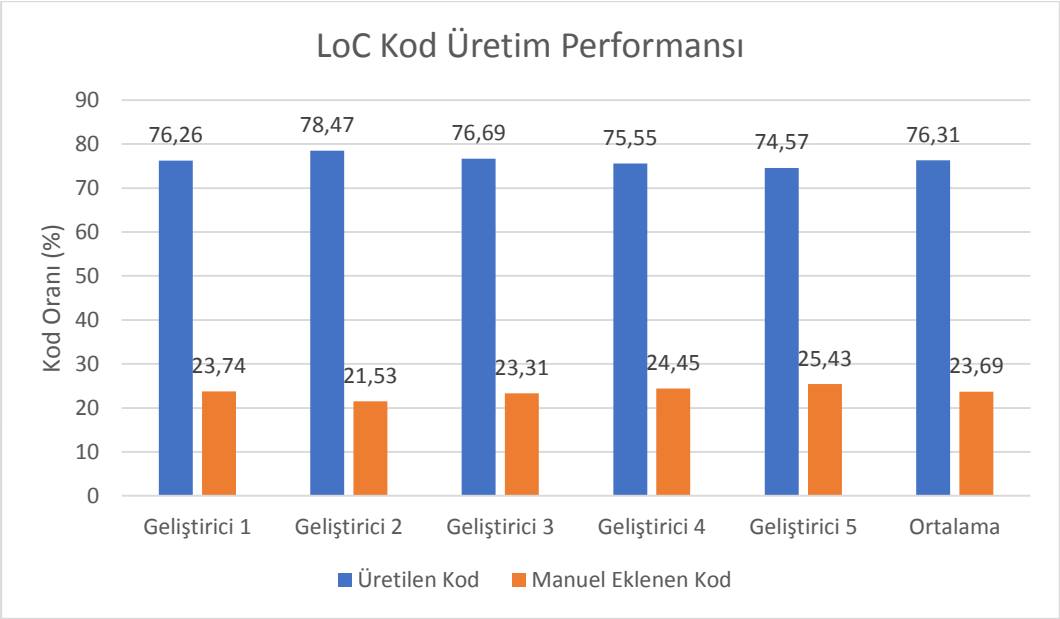
8.2 Sonuçlar ve Tartışma

DSML4DT'nin kullanımı, hem kod üretimi hem de tüm geliştirme sürecinde kaydedilen zamanlar üzerinden nicel olarak analiz edilmiştir.

Kod Üretim Performansı: DSML4DT'nin üretken yönü göz önünde bulundurulduğunda, temel yapılar DT modeli ile üretilmiştir. Üretim performansını hesaplamak için, otomatik olarak oluşturulan kod ile kodun tamamlanması için eklenen delta kodu arasındaki karşılaştırmalar yapılmıştır. Performans değerlendirmesi, otomatik olarak üretilen ve elle geliştirilen yazılım kodlarının yüzdesi karşılaştırılarak yerine getirilmiştir. Sürücü bilgisayarındaki kullanılan DT dosyalarının Kod Satır Sayısı (ing. Lines of Codes) (LoC) performansı için deney ortalamaları ve tüm sistemlerin ortalaması hesaplanmıştır. Bu sonuçlar, karşılaştırma ve analizi kolaylaştırmak için Şekil 8.1'de sunulmuştur. İlk olarak, tüm geliştiriciler kod geliştirmek için DSML4DT'yi kullanır ve daha sonra kodu elle tamamlar. Genel Ortalama, tüm vaka çalışmalarından elde edilen LoC ortalamasıdır. Örnek çalışma cihazında ihtiyaç duyulan DT dosyaları Şekil 8.1'de görüldüğü gibi üretilmiştir. Burada yüzde olarak görülebilen LoC, geliştirici bazlı ve genel ortalama olarak verilmiştir.

Geliştiricilerin modellerinden otomatik üretilen kod ortalaması 900,8 satırdır. Bu satırların üzerine geliştiricilerin eklediği kod satırı ortalaması ise 279,6'dır. Eklenen kod ve otomatik üretilen kodun toplamının ortamalası böylelikle 1180,4 olmaktadır. Burada yapılan hesaplamalar üzerinden elde edilen ve Şekil 8.1'de verilen sonuçlara göre, ortalama üretilen LoC oranı %76,31'dir. Bu oran değerlendirmeye katılan DT geliştiricilerine göre %74 ile %78 arasında değişmektedir. Bu varyasyonun %4 civarında oldukça küçük olduğu görülebilmektedir. Bu ufak farklılıklar geliştiricilerin DT alanına olan hakimiyet değişikliklerine bağlanabilir. Geliştiriciler, genel olarak, cihazda daha fazla deneyime sahiplerse daha iyi LoC oranlarına sahiptir. Bu farklılıkların etkisini

azaltmak için geliştirici sayısı mümkün olduğunca yüksek tutulmuştur. Böylece, bu deneylerin LoC ortalaması %76,31 olarak hesaplanmıştır. Gömülü sistemler gibi çok geniş varyasyonlara sahip mimarilerin bulunduğu alanda bu LoC oranı oldukça yüksektir.



Şekil 8.1 DSML4DT LoC üretim performansı

Otomatik kod üretiminden sonra elle eklenen %23,69 oranındaki kodun azaltılması için metamodelün genişletilmesi düşünülebilir. Metalmodelde bulunan öğelerin sayısı ve kapsamı gömülü sistemlerin daha büyük alanını kapsayacak şekilde arttırılabilir. Ancak bu durumda zaten oldukça geniş ve karmaşık olan metamodelin daha karmaşık hale gelmesi ve yazılım geliştiriciler tarafından kullanımının güçleşmesi söz konusudur. Sonuçta metamodel tasarlanırken bu dengenin iyi kurulması önem arz etmektedir.

Geliştirme Zamanı Performansı: DSML4DT'yi kullanmanın geliştirme zamanına olan etkilerini görmek için zaman ölçümleri ve analizleri yapılmıştır. Geliştirme çabaları sırasında kaydedilen zamanları ölçmek için, DSML4DT aracı kullanılan deneyler (Deney A) ve herhangi bir alana özel araç kullanılmayan deneyler yapılmıştır (Deney B). Deney sonuçları karşılaştırılmış ve analiz edilmiştir. Deney B sürecinde, geliştiriciler klasik, elle yazılım geliştirme yapmışlardır. Geliştiriciler için kaydedilen geliştirme süreleri, Bölüm 8.1'de tartışılan tüm DT geliştirme adımlarını içermektedir. Bu adımların için ortalama süre, her iki deney için ayrı ayrı hesaplanmıştır. Deney A ve B'den elde edilen

sonuçlar, kolay karşılaştırılabilmesi için Şekil 8.2'de sunulmuştur. Şekil 8.2'de verilen sonuçlara dayalı olarak tüm geliştirme aşamaları için aşağıdaki yorumlar yapılabilmektedir.

Analiz - Sistemin analizi, gelişim sürecinin bir parçasıdır. Analiz adımı herhangi bir araç veya platforma bağlı değildir. Bu sebepten dolayı, Deney A ve Deney B için neredeyse benzer analiz süreleri vardır. Aslında bu adım sadece gömülü cihazın DT yapısının karmaşıklığına bağlıdır. Bu adım geliştirme dilinden bağımsızdır. Bu nedenle, zamanlar arasındaki fark göz ardı edilebilir. Zaten süreler birbirine oldukça yakın ölçülmüştür.

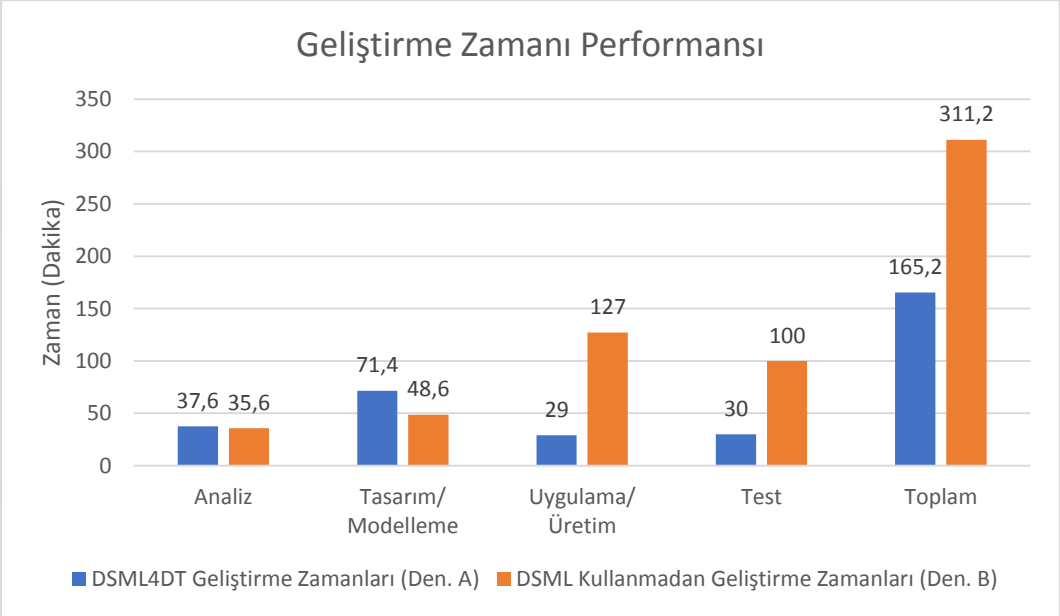
Tasarım/Modelleme – Bu adımda süreler birbirine yakın gözükse de Deney A'daki zaman biraz daha fazladır. DSML4DT, daha ayrıntılı elemanlar ve üretim sırasında kullanılabilecek niteliklerini içerecek şekilde tasarım zamanında DT alanına özgü konseptlerle çalışır. Ayrıca, DSML4DT, tasarımcıların, daha sonra bunları düzeltmek için daha fazla zaman alan anlamsal hataları önleyebilmelerini sağlayan statik semantik denetimlere sahiptir. Bu gibi nedenlerle, DSML4DT araçları ile modelleme, tasarım süresini biraz daha uzatır. Bu tür bir ayrıntılı modelleme Deney B geliştiricileri tarafından yapılmamıştır. Geçen süre, DSML4DT kullanıcılarının otomatik olarak oluşturulacak daha ayrıntılı ve doğru kodlar elde etmelerine yönelik bir yatırım olarak görülebilmektedir.

Uygulama/Üretim – Bu adım, DSML4DT performansını klasik yazılım geliştirme süreci ile karşılaştırmak için en iyi adımdır. Deney A'da DT kodu otomatik olarak elde edilir. Deney B'de geliştiriciler tasarım için gerekli kodları elle yazmışlardır. DSML4DT, ayrıntılı modellemeden elde edilen ürünleri kullanarak otomatik DT kodları üretmeyi başarmıştır. Ek kodların da Deney A'da tamamlanması ile süreler karşılaştırılmıştır. DSML4DT kullanılarak yapılan geliştirme süresinin 4 katından daha fazla süre Deney B'de görülmüştür. Bu adım DSML4DT'nin en başarılı olduğu adımdır.

Test – Üretim adımında olduğu gibi, geliştirme testi süresi DSML4DT tarafından önemli ölçüde azaltılmıştır. Test/hata ayıklama adımında, tüm geliştiriciler programların sözdizimsel ve anlamsal hatalarını bulmaya çalışmışlardır. Deney B'de geliştiriciler tüm kodda hata aramışlardır. Deney A'da ise geliştiriciler sadece oluşturulan koddan çok daha az olan delta kodundaki problemler ile uğraşmışlardır. Oluşturulan kod, son kodun çoğunu oluşturduğundan ve DSML4DT kısıtlama denetimleri tarafından doğrulandığından, yeniden

doğrulanması gerekmeyen, hatasız ve neredeyse hazır mimari kodlardan oluşmaktadır. Bu nedenle, Deney A'da hataları bulmak için gereken süre Deney B'deki süreden 3,3 kat daha azdır.

Toplam Zaman – Deney A'daki toplam ortalama süre tüm vaka çalışmaları değerlendirildiğinde 165 dakikadır. Deney B'deki toplam ortalama süre yaklaşık 311 dakikadır. Böylece DSML4DT kullanımı ile toplam geliştirme süresinin nereyse yarıya düştüğü bir sürücü bilgisayarı cihazında, 5 farklı geliştirici ile yapılan çalışmalar sonucunda gösterilmiştir.



Şekil 8.2 DSML4DT geliştirme zamanı performansı

9. SONUÇ

Gömülü sistemlerde kullanılan DT yazılımlarının geliştirilmesi için bir DSML bu tezde sunulmuştur. Bu çalışmada geliştirilen dil, grafik modelleme ve otomatik kod oluşturma özelliklerini içermektedir. Bu çalışmanın değerlendirme aşamasında DT geliştiricileri ile çalışma yapılmıştır. Yapılan zaman ölçümlerinden ve LoC hesaplamalarından sonra elde edilen değerlendirmeye göre, dilin kullanılmasının DT yazılım uygulamaları için gerekli kodun büyük bir kısmının otomatik olarak elde edilmesine izin verdiği belirlenmiştir. Deneysel çalışmalar, dilin kullanımının gelişme sürecini kısalttığını göstermiştir. Örnek durum çalışması sonuçlarına göre DT yazılım geliştirme süresinin DSML4DT kıullanılarak klasik DT geliştirme yöntemine göre yarı yarıya azaldığı görülmüştür.

DT yazılım geliştiricileri şu anda DT yazılımını otomatik olarak geliştirmek için model tabanlı araçlara sahip değillerdir. Geliştiriciler, kendi özel sözdizimi ve mantığı olan DT yazılımını geliştirmek için klasik manuel kod geliştirme yöntemlerini kullanmaktadırlar. Zaten karmaşık olan gömülü sistemlerde, DT yazılımı geliştirme de oldukça zor olabilmektedir. Bu nedenlerden dolayı, bu dilin, gelişim süresini kısaltan ve süreci basitleştiren kullanımının yaygınlaşabileceğine inanılmaktadır.

DSML4DT'nin iyileştirilmesi ve geliştirilmesi için ileriye yönelik yeni çalışmalar yapılabilir. Bu kapsamda, farklı mikroişlemci içeren gömülü sistem platformlarında ve cihazlarında DT yazılımının uygulamalarının genişletilmesi üzerinde çalışmalar uygundur. Linux ve Android'in farklı sürümleri için DT çalışmaları da bu kapsama girebilir. Farklı işletim sistemleri ile uyumlu olabilecek model tabanlı aygıt sürücüsü kodlarının oluşturulması üzerinde yeni çalışmalar yapılabilir.

Bu tezde tek bir cihaz ile yapılan vaka çalışması daha fazla geliştirilebilir. Yeni vaka çalışmaları kullanılarak DSML4DT'nin geliştirme performansı çalışmaları hakkında daha ayrıntılı bilgi toplanması imkânı bulunmaktadır. Bu vaka çalışmaları üzerinde anketler de kullanılarak değerlendirme genişletilebilir ve nitekliksel değerlendirmeler de yapılabilir.

KAYNAKLAR DİZİNİ

Acceleo, 2018, Acceleo Model to Text Language, https://www.eclipse.org/acceleo/, (Erişim tarihi: 14 Mayıs 2018).

Arslan, S., Türk E. ve Kardaş, G., 2017, Gömülü Sistem Yazılımlarının Geliştirilmesinde Aygıt Ağacı Yapısının Kullanılmasına Yönelik bir Çalışma, 2. Uluslararası Bilgisayar Bilimleri ve Mühendisliği Konferansı (UBMK), 2017, p: 882 – 887, IEEE Conference Publications, DOI: 10.1109/UBMK.2017.8093472.

Bryant B. R., Gray, J., Mernik, M., Clarke, P. J., France, R. B. and Karsai, G., 2011, Challenges and Directions in Formalizing the Semantics of Modeling Languages, Computer Science and Information Systems, 8(2), pp. 225-253.

Challenger, M., Kardas, G. and Tekinerdogan, B., 2016, A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems, Software Quality Journal, 24(3):755-795.

Chen, H., Godet-Bar, G., Rousseau, F. and Petrot, F., 2011, Me3D: A model-driven methodology expediting embedded device driver development, In proceedings of 22th IEEE International Symposium on Rapid System Prototyping, pp. 171-177.

Chen, H., Godet-Bar, G., Rousseau, F. and Petrot, F., 2014, Device driver generation targeting multiple operating systems using a model-driven methodology, In proceedings of 25th IEEE International Symposium on Rapid System Prototyping, pp. 30-36.

Clark, T., Evans, A., Sammut, P. and Willans, J., 2004, Language Driven Development and MDA. MDA Journal, pp. 2-13.

Cuadrado, S. J. and Molina, G. J., 2007, Building Domain-Specific Languages for Model-Driven Development. IEEE Software, 24(5), pp. 48-56.

Devicetree Community, 2016, The Devicetree Specification, https://www.devicetree.org/, (Erişim tarihi: 14 Mayıs 2018).

Devigne, C., Brejon, J.-B., Meunier, Q., L. and Wajsbürt, F., 2017, Executing secured virtual machines within a manycore architecture, Microprocessors and Microsystems, 48: 21-35.

EMF, 2008, Eclipse Modelling Framework, https://www.eclipse.org/modeling/emf/ (Erişim tarihi: 14 Mayıs 2018).

KAYNAKLAR DİZİNİ (devam)

Freemind, Freemind, 2002. available at: http://freemind.sourceforge.net/wiki/index.php/Main_Page (Erişim tarihi: 14 Mayıs 2018).

Fowler, M., 2011, Domain-specific Languages. Addison-Wesley Professional, 640p.

Gajski, D. D., Vahid, F., Narayan, S. and Gong, J., 1994, Specification and Design of Embedded Systems (1st Edition), Prentice Hall.

GME, Ledeczi A., Maroti M., Bakay A., Karsai G., Garrett J., Thomason C., Nordstrom G., Sprinkle J.,Volgyesi P., 2001, The Generic Modeling Environment, In Proceedings of the Workshop on Intelligent Signal Processing, http://www.isis.vanderbilt.edu/Projects/gme/ (Erişim tarihi: 14 Mayıs 2018).

Gray, J., Tolvanen, J-P., Kelly, S., Gokhale, A., Neema, S. and Sprinkle, J., 2007, Domain-Specific Modeling. In Fishwick (Ed): Handbook of Dynamic System Modeling, CRC Press, pp. 1-7.

IEEE Standard 1275-1994, 1994, IEEE Standard 1275-1994 for Boot (Initialization Configuration) Firmware: Core Requirements and Practices.

IEEE Standard 1685-2014, 2014, IEEE Standard 1685-2014 for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows.

Intel, 2006, PXA27x Processor Family Developer's Manual.

Jassi, M., Sharif, U., Müller-Gritschneder, D. and Schlichtmann, U., 2016. Hardware-accelerated software library drivers generation for IP-centric SoC designs, In proceedings of the 26th Great Lakes Symposium on VLSI, pp. 287-292.

Kamal R., 2009, Embedded Systems: Architecture, Programming and Design (2nd Edition), New Delhi: McGraw-Hill Education.

Katayama, T., Saisho, K. and Fukuda, A., 2000, Prototype of the device driver generation system for UNIX-like operating systems, In proceedings of the International Symposium on Principles of Software Evolution.

Kelly, S., and Tolvanen, J-P., 2008, Domain-Specific Modeling: Enabling Full Code Generation. John Willey & Sons, Inc., New Jersey, USA.

KAYNAKLAR DİZİNİ (devam)

Kent Kart, 2018, Kent Kart Ege Elektronik A.Ş., https://www.kentkart.com/ (Erişim tarihi: 14 Mayıs 2018).

King, M., Dave, N. and Arvind, 2012, Automatic generation of hardware/software interfaces, ACM SIGPLAN Notices, 47(4): 325-336.

Lee, E. A. and Seshia, S. A., 2016, Introduction to Embedded Systems: A Cyber-Physical Systems Approach (1st Edition), MIT Press.

Lecomte, S., Guillouard, S., Moy, M., Leray, P. and Soulard, P., 2011, A co-design methodology based on model driven architecture for real time embedded systems, Mathematical and Computer Modelling, 53(3-4): 471-484.

Likely, G. and Boyer, J., 2008, A Symphony of Flavours: Using the device tree to describe embedded hardware, In proceedings of 2008 Linux Symposium, Volume Two, pp. 27-37.

Liu, S.-H., Cardenas, A., Mernik, M., Bryant, B. R., Gray, J. and Xiong, X., 2012, Introducing domain-specific language implementation using web service-oriented technologies, Multiagent and Grid Systems, 8(1): 19-44pp.

Mernik, M. (Ed), 2013, Formal and Practical Aspects of Domain-Specific Languages: Recent Developments, IGI Global.

Mernik, M., Heering, J. and Sloane, A. M., 2005, When and how to develop domain-specific languages, ACM Computing Surveys, 37(4): 316-344pp.

MetaCase, 1995, MetaCase, MetaEdit+ Domain-Specific Modeling (DSM) environment, http://www.metacase.com/products.html (Erişim tarihi: 14 Mayıs 2018).

Microsoft, 2005, Microsoft DSL Tools, https://msdn.microsoft.com/en-us/library/bb126327.aspx (Erişim tarihi: 14 Mayıs 2018).

Nicolescu, G. and Mosterman, P. J., 2010. Model-based Design for Embedded Systems. United States of America: Taylor & Francis Group.

Nikkel, B., 2016, NVM express drives and digital forensics, Digital Investigation, 16: 38-45.

KAYNAKLAR DİZİNİ (devam)

NXP Semiconductor, 2013, i.MX 6Dual/6Quad Applications Processor Reference Manual, IMX6DQRM Rev. 1, 04/2013.

OMG, 2003, Model Driven Architecture (MDA) Specification, Object Management Group. http://www.omg.org/mda/, (Erişim tarihi: 14 Mayıs 2018).

OMG, 2017, Object Management Group, http://www.omg.org/, (Erişim tarihi: 14 Mayıs 2018).

Petazonni, T., 2013, Device Tree, https://events.linuxfoundation.org/ sites/events/files/slides/petazzoni-device-tree.pdf, (Erişim tarihi: 14 Mayıs 2018).

Rocketboards, 2016, "Golden System Reference Design (GSRD)". https://rocketboards.org/foswiki/view/Documentation/DeviceTreeGenerator (Erişim tarihi: 14 Mayıs 2018).

Schmidt, D.C., 2006, Guest Editor's Introduction: Model-Driven Engineering. IEEE Computer, 39(2), pp. 25-31.

Selic, B., 2003, The pragmatics of model-driven development, IEEE Software, 20: 19-25pp.

Simmonds, C., 2015, Mastering Embedded Linux Programming, Birmingham, UK: Packt Publishing.

Sprinkle, J., Mernik, M., Tolvanen, J.-P. and Spinellis, D., 2009, Guest Editors' Introduction: What Kinds of Nails Need a Domain-Specific Hammer?, IEEE Software, 26(4), pp. 15-18.

Strembeck, M. and Zdun, U., 2009, An approach for the systematic development of domain-specific languages, Software-Practice & Experience, 39(15): 1253-1292pp.

The Eclipse Foundation, 2006, Graphical Modeling Framework (GMF), http://www.eclipse.org/modeling/gmp/ (Erişim tarihi: 14 Mayıs 2018).

The Sirius Project, 2013, The Eclipse Sirius Modelling Project, 2013, http://www.eclipse.org/sirius/ (Erişim tarihi: 14 Mayıs 2018).

van Deursen, A., Klint, P. and Visser, J., 2000, Domain-specific languages: An annotated bibliography, ACM SIGPLAN Notices 35(6): 26-36pp.

KAYNAKLAR DİZİNİ (devam)

Varanda-Pereira, M. J., Mernik, M., Da-Cruz, D. and Henriques, P. R., 2008, Program comprehension for domain-specific languages, Computer Science and Information Systems, 5(2), 1-17pp.

ÖZGEÇMİŞ

Adı Soyadı : Sadık Arslan

Doğum tarihi : 15.06.1985

Doğum yeri : Kırcali

E-mail : sadikarslan1985@yahoo.com

Eğitim :

- 20013–2014 Maltepe Üniversitesi, Sosyal Bilimler Enstitüsü, İşletme Yüksek Lisansı, 3.5/4
- 2003–2008 Ege Üniversitesi Elektronik Mühendisliği Bölüm, 2.88/4
- 1999-2003 Konak Anadolu Lisesi, 4,24/5

İş deneyimi :

- Kent Kart Ege Elektronik A.Ş., Gömülü Sistem Yazılım Geliştirme Müh., 2012 – devam ediyor.
- Vestel Elektronik A.Ş., Program Yöneticisi, 2011 – 2012.
- Vestel Elektronik A.Ş., Sistem ve Donanım Tasarım Müh., 2008 – 2011.

Yayınlar :

- Sadık Arslan, Moharram Challenger, Orhan Dagdeviren, Wireless sensor network based fire detection system for libraries, International Conference on Computer Science and Eng., 2017.

- Sadık Arslan, Mustafa Gündüzalp, Implementation of a Sodimm Interfaced Embedded and Modular Controller Board, Journal of Multidisciplinary Engineering Science and Technology, Vol. 4 Issue 6, June 2017,

- Sadık Arslan, Mustafa Gündüzalp, An Embedded Control Card Design Including a Microprocessor, Electric-Electronics and Computer Symposium, May 2016

- Ercüment Türk, Sadık Arslan, "Toplu Taşımada Düşük Güçlü Bluetooth ile Konum Bazlı Ücretlendirme", 25. Sinyal İşleme Ve İletişim Uygulamaları Kurultayı, 2016

- Ercüment Türk, Aydoğan Savran, Sadık Arslan, "Android Tabanlı Mobil Ödeme ve Araç İçi Kontrol Merkezi", 2016 Electrical, Electronics and Biomedical Engineering (ELECO), 466 – 470

- Ercüment Türk, Sadık Arslan, "Gömülü sistemlere Android İşletim Sistemi Uyumlandırma", 1. Uluslararası Akdeniz Bilim ve Mühendislik Kongresi, 2016, 1783-1790

- Ercüment Türk, Mustafa Gündüzalp, Sadık Arslan, "Bir Toplu Taşıma Sisteminde Yardımlı Küresel Yer Belirleme Sitemi ve Seyrüsefer Hesabı Yöntemlerinin Uygulanması", 1. Uluslararası Akdeniz Bilim ve Mühendislik Kongresi, 2016, 1540 – 1564

- Sadık Arslan, Mustafa Gündüzalp, Ercüment Türk, "Gömülü Bir Sistem İçin Bir Çoklu Ortam Uygulaması", 2016 Electrical, Electronics and Biomedical Engineering (ELECO), 2016, 189 - 193

- Sadık Arslan, Ercüment Türk, "Gömülü Sistem Bir Ürün İçin Veri Tabanı Kullanılarak Montaj, Sipariş ve Fiyatlandırma Uygulaması", 2016 Electrical, Electronics and Biomedical Engineering (ELECO), 2016, 705- 709

- Sadık Arslan, Ercüment Türk, "Toplu Tama Sistemleri İçin Bir Araç İçi Görüntüleme Uygulaması", 2016 Electrical, Electronics and Biomedical Engineering (ELECO), 2016, 461- 465

- Sadık Arslan, Ercüment Türk, "Gömülü Sistem Bir Araç Takip Sistemi Uygulaması", 2016 Electrical, Electronics and Biomedical Engineering (ELECO), 2016, 447- 451

- Sadık Arslan, Mustafa Gündüzalp, Ercüment Türk, "Ücret toplama sistemlerinde kullanılan gömülü bir cihaz için telefon rehberi ve telefon görüşme sistemi uygulaması", 1. Uluslararası Akdeniz Bilim ve Mühendislik Kongresi, 2016, 1176 – 1182

- Sadık Arslan, Fatih Yücalar, Bug Tracking and Project Management System Application in an Electronic Design Company, Pamukkale University Journal of Engineering Sciences, vol:2 (2015)

- Sadık Arslan, Veli Demirel, İbrahim Kuru, A Public Transport Fare, Collection System with Smart Phone Based NFC Interface, International Journal of Electronics and Electrical Engineering, Vol. 4, No. 3, June 2016, pp 258-263.

- Sadık Arslan, Fatih Yücalar, An Application of Management Information System for R&D Department, 2015 Academic Computing Conference, 04-06 Februay 2015

- Sadık Arslan, ARM Microcontroller Based Embedded Battery Backup and Remote Monitoring System, Gaziosmanpaşa Journal of Scientific Research, vol:9(2014), pp: 1-12.

- Sadık Arslan, Read/Write Mode Operating Distance Optimization for 13.56MHz Contactless IC Cards, 8th International Conference on Electrical and Electronics Engineering, November 28-30 2013.

- Sadık Arslan, Sayısal Televizyonlar İçin Bir Görüntü, Ses ve Fotoğraf İşleme Uygulaması, The sixth International Advanced Technologies Symposium, 16th - 18th May 2011.

EKLER

Ek 1 Acceleo Kodu

Ek 2 Üretilen DT Kodu

Ek 3 Bilgilendirilmiş Onam Formu

Ek 1 Acceleo Kodu

```
[comment encoding = UTF-8 /]
[module generate('http://www.example.org/devicetree_viewpoint')]


[template public generateElement(aroot : root)]
[comment @main/]
generateElement(index : Integer)
[file ('imx6dl.dts', false)]
/*
 *Copyright 2018 Kent Kart Ege Elektronik A.S.
 *
 *This code contains iMX6dl based device tree structure for driver computer board
 *
 */

#include <dt-bindings/interrupt-controller/irq.h>
#include "imx6dl-pinfunc.h"
#include "imx6qdl.dtsi"
#include <dt-bindings/clock/imx6qdl-clock.h>
#include <dt-bindings/gpio/gpio.h>
#include <dt-bindings/interrupt-controller/arm-gic.h>
#include "skeleton.dtsi"
#include <dt-bindings/input/input.h>

/{
        model = [aroot.model/];
        compatible = [aroot.compatible/];

        [aroot.cpus.name/]{
          address-cells = <[aroot.cpus.address_cells/]>;
          size-cells = <[aroot.cpus.size_cells/]>;

          [for (Sequence{1, 2, 3, 4, 5, 6, 7, 8})]
          [if (aroot.cpus.cpu.name->at(i)->notEmpty())]
          [aroot.cpus.cpu.name->at(i)/]{
                  compatible = [aroot.cpus.cpu.compatible->at(i)/];
                  device_type = [aroot.cpus.cpu.device_type->at(i)/];
                  reg = <[aroot.cpus.cpu.reg->at(i)/]>;
                  next-level-cache = <[aroot.cpus.cpu.next_level_cache->at(i)/]>;
                  [if (aroot.cpus.cpu.operating_points->at(i)->notEmpty())]
                  operating-points = <[aroot.cpus.cpu.operating_points->at(i)/]>;
                  [/if]
                  [if (aroot.cpus.cpu.soc_operating_points->at(i)->notEmpty())]
                  fsl,soc-operating-points = <[aroot.cpus.cpu.soc_operating_points->at(i)/]>;
                  [/if]
                  [if (aroot.cpus.cpu.clock_latency->at(i)->notEmpty())]
                  clock-latency = <[aroot.cpus.cpu.clock_latency->at(i)/]>;
                  [/if]
                  [if (aroot.cpus.cpu.clocks->at(i)->notEmpty())]
                  clocks = [aroot.cpus.cpu.clocks->at(i)/];
                  [/if]
                  [if (aroot.cpus.cpu.clock_names->at(i)->notEmpty())]
                  clock-names = [aroot.cpus.cpu.clock_names->at(i)/];
                  [/if]
                  };

          [/if]
          [/for]
        };

        [aroot.memory.name/]{
          reg = <[aroot.memory.reg/]>;
        };

        [if (aroot.chosen.bootargs->notEmpty())]
        [aroot.chosen.name/]{
          bootargs = [aroot.chosen.bootargs/];
        };
        [/if]

        [aroot.aliases.name/]{
```

```
    gpio_s = [aroot.aliases.gpio_s/];
    i2c_s = [aroot.aliases.i2c_s/];
    mmc_s = [aroot.aliases.mmc_s/];
    serial_s = [aroot.aliases.serial_s/];
    spi_s = [aroot.aliases.spi_s/];
    usbphy_s = [aroot.aliases.usbphy_s/];
    mxcfb_s = [aroot.aliases.mxcfb_s/];
};

[if (aroot.gpio_keys.name->notEmpty())]
[aroot.gpio_keys.name/]{
    compatible = [aroot.gpio_keys.compatible/];
    pinctrl-names = [aroot.gpio_keys.pinctrl_names/];
    pinctrl-0 = <[aroot.gpio_keys.pinctrl_0/]>;

    [if (aroot.gpio_keys.power.name->notEmpty())]
    [aroot.gpio_keys.power.name/]{
                label = [aroot.gpio_keys.power.label/];
                gpios = <[aroot.gpio_keys.power.gpios/]>;
                gpio-key[aroot.gpio_keys.power.gpio_key/];
                linux,code = <[aroot.gpio_keys.power.linux_code/]>;
    };
    [/if]
};
[/if]

[if (aroot.sound.name->notEmpty())]
[aroot.sound.name/]{
    compatible = [aroot.sound.compatible/];
    model = [aroot.sound.model/];
    cpu-dai = <[aroot.sound.cpu_dai/]>;
    audio-codec = <[aroot.sound.audio_codec/]>;
    audio-routing = [aroot.sound.audio_routing/];
    mux-int-port = <[aroot.sound.mux_int_port/]>;
    mux-ext-port = <[aroot.sound.mux_ext_port/]>;
};
[/if]

[if (aroot.sound_spdif.name->notEmpty())]
[aroot.sound_spdif.name/]{
    compatible = [aroot.sound_spdif.compatible/];
    model = [aroot.sound_spdif.model/];
    spdif-controller = <[aroot.sound_spdif.spdif_controller/]>;
    spdif-in[aroot.sound_spdif.spdif_in/];
    spdif-out[aroot.sound_spdif.spdif_out/];
};
[/if]

[if (aroot.sound_hdmi.name->notEmpty())]
[aroot.sound_hdmi.name/]{
    compatible = [aroot.sound_hdmi.compatible/];
    model = [aroot.sound_hdmi.model/];
    spdif-controller = <[aroot.sound_hdmi.hdmi_controller/]>;
};
[/if]

[if (aroot.backlight.name->notEmpty())]
[aroot.backlight.name/]{
    compatible = [aroot.backlight.compatible/];
    pwms = <[aroot.backlight.pwms/]>;
    brightness-levels = <[aroot.backlight.brigtness_levels/]>;
    default-brightness-level = <[aroot.backlight.default_brigtness_level/]>;
    status = [aroot.backlight.status/];
};
[/if]

[if (aroot.battery.name->notEmpty())]
[aroot.battery.name/]{
    offset-charger = [aroot.battery.offset_charger/];
    offset-discharger = <[aroot.battery.offset_discharger/]>;
    offset-usb-charger = <[aroot.battery.offset_usb_charger/]>;
};
[/if]

[if (aroot.lcd.name->notEmpty())]
[aroot.lcd.name/]{
```

```
    compatible = [aroot.lcd.compatible/];
    ipu_id = <[aroot.lcd.ipu_id/]>;
    disp_id = <[aroot.lcd.disp_id/]>;
    default_ifmt = [aroot.lcd.default_ifmt/];
    pinctrl-names = [aroot.lcd.pinctrl_names/];
    pinctrl-0 = <[aroot.lcd.pinctrl_0/]>;
    status = [aroot.lcd.status/];
};
[/if]

[if (aroot.interrupt_controller.name->notEmpty())]
[aroot.interrupt_controller.name/]{
    compatible = [aroot.interrupt_controller.compatible/];
    interrupt-cells = <[aroot.interrupt_controller.interrupt_cells/]>;
    address-cells = <[aroot.interrupt_controller.address_cells/]>;
    size-cells = <[aroot.interrupt_controller.size_cells/]>;
    interrupt-controller[aroot.interrupt_controller.interrupt_controller/];
    reg = [aroot.interrupt_controller.reg/];
};
[/if]

[aroot.soc.name/]{
    address-cells = <[aroot.soc.address_cells/]>;
    size-cells = <[aroot.soc.size_cells/]>;
    compatible = [aroot.soc.compatible/];
    interrupt-parent = <[aroot.soc.interrupt_parent/]>;
    ranges[aroot.soc.ranges/];

    [if (aroot.soc.gpmi_nand.name->notEmpty())]
    [aroot.soc.gpmi_nand.name/]{
            compatible = [aroot.soc.gpmi_nand.compatible/];
            address-cells = <[aroot.soc.gpmi_nand.address_cells/]>;
            size-cells = <[aroot.soc.gpmi_nand.size_cells/]>;
            reg = [aroot.soc.gpmi_nand.reg/];
            reg-names = [aroot.soc.gpmi_nand.reg_names/];
            interrupts = <[aroot.soc.gpmi_nand.interrupts/]>;
            interrupt-names = [aroot.soc.gpmi_nand.interrupt_names/];
            clocks = [aroot.soc.gpmi_nand.clocks/];
            clock-names = [aroot.soc.gpmi_nand.clock_names/];
            dmas = <[aroot.soc.gpmi_nand.dmas/]>;
            dma-names = [aroot.soc.gpmi_nand.dma_names/];
            status = [aroot.soc.gpmi_nand.status/];
    };
    [/if]

    [if (aroot.soc.ipu.name->notEmpty())]
    [aroot.soc.ipu.name/]{
            compatible = [aroot.soc.ipu.compatible/];
            reg = <[aroot.soc.ipu.reg/]>;
            interrupts = [aroot.soc.ipu.interrupts/];
            clocks = [aroot.soc.ipu.clocks/];
            clock-names = [aroot.soc.ipu.clock_names/];
            resets = <[aroot.soc.ipu.resets/]>;
            bypass_reset = <[aroot.soc.ipu.bypass_reset/]>;
    };
    [/if]

    [if (aroot.soc.hdmi_core.name->notEmpty())]
    [aroot.soc.hdmi_core.name/]{
            compatible = [aroot.soc.hdmi_core.compatible/];
            reg = <[aroot.soc.hdmi_core.reg/]>;
            clocks = [aroot.soc.hdmi_core.clocks/];
            clock-names = [aroot.soc.hdmi_core.clock_names/];
            status = [aroot.soc.hdmi_core.status/];
    };
    [/if]

    [if (aroot.soc.hdmi_video.name->notEmpty())]
    [aroot.soc.hdmi_video.name/]{
            compatible = [aroot.soc.hdmi_video.compatible/];
            reg = <[aroot.soc.hdmi_video.reg/]>;
            reg-names = [aroot.soc.hdmi_video.reg_names/];
            interrupts = <[aroot.soc.hdmi_video.interrupts/]>;
            clocks = [aroot.soc.hdmi_video.clocks/];
            clock-names = [aroot.soc.hdmi_video.clock_names/];
            status = [aroot.soc.hdmi_video.status/];
```

```
};
[/if]

[if (aroot.soc.hdmi_audio.name->notEmpty())]
[aroot.soc.hdmi_audio.name/]{
        compatible = [aroot.soc.hdmi_audio.compatible/];
        clocks = [aroot.soc.hdmi_audio.clocks/];
        clock-names = [aroot.soc.hdmi_audio.clock_names/];
        dmas = <[aroot.soc.hdmi_audio.dmas/]>;
        dma-names = [aroot.soc.hdmi_audio.dma_names/];
        status = [aroot.soc.hdmi_audio.status/];
};
[/if]

[if (aroot.soc.hdmi_cec.name->notEmpty())]
[aroot.soc.hdmi_cec.name/]{
        compatible = [aroot.soc.hdmi_cec.compatible/];
        interrupts = <[aroot.soc.hdmi_cec.interrupts/]>;
        status = [aroot.soc.hdmi_cec.status/];
};
[/if]

[if (aroot.soc.gpu.name->notEmpty())]
[aroot.soc.gpu.name/]{
        compatible = [aroot.soc.gpu.compatible/];
        reg = [aroot.soc.gpu.reg/];
        reg-names = [aroot.soc.gpu.reg_names/];
        interrupts = [aroot.soc.gpu.interrupts/];
        interrupt-names = [aroot.soc.gpu.interrupt_names/];
        clocks = [aroot.soc.gpu.clocks/];
        clock-names = [aroot.soc.gpu.clock_names/];
        resets = [aroot.soc.gpu.resets/];
        reset-names = [aroot.soc.gpu.reset_names/];
        power-domains = [aroot.soc.gpu.power_domains/];
};
[/if]

[if (aroot.soc.l2_cache.name->notEmpty())]
[aroot.soc.l2_cache.name/]{
        compatible = [aroot.soc.l2_cache.compatible/];
        regs = <[aroot.soc.l2_cache.reg/]>;
        interrupts = <[aroot.soc.l2_cache.interrupts/]>;
        cache-unified[aroot.soc.l2_cache.cache_inified/];
        cache-level = <[aroot.soc.l2_cache.cache_level/]>;
        arm,tag-latecy = <[aroot.soc.l2_cache.arm_tag_latency/]>;
        arm,data-latecy = <[aroot.soc.l2_cache.arm_data_latency/]>;
};
[/if]

[if (aroot.soc.timer.name->notEmpty())]
[aroot.soc.timer.name/]{
        compatible = [aroot.soc.timer.compatible/];
        reg = <[aroot.soc.timer.reg/]>;
        interrupts = <[aroot.soc.timer.interrupts/]>;
        clocks = <[aroot.soc.timer.clocks/]>;
};
[/if]

[if (aroot.soc.pcie.name->notEmpty())]
[aroot.soc.pcie.name/]{
        compatible = [aroot.soc.pcie.compatible/];
        reg = [aroot.soc.pcie.reg/];
        reg-names = [aroot.soc.pcie.reg_names/];
        address-cells = <[aroot.soc.pcie.address_cells/]>;
        size-cells = <[aroot.soc.pcie.size_cells/]>;
        device_type = [aroot.soc.pcie.device_type/];
        ranges = [aroot.soc.pcie.ranges/];
        num-lanes = <[aroot.soc.pcie.num_lanes/]>;
        interrupts = [aroot.soc.pcie.interrupts/];
        interrupt-names = [aroot.soc.pcie.interrupt_names/];
        interrupt-cells = <[aroot.soc.pcie.interrupt_cells/]>;
        interrupt-map-mask = <[aroot.soc.pcie.interrupt_map_mask/]>;
        interrupt-map = [aroot.soc.pcie.interrupt_map/];
        clocks = [aroot.soc.pcie.clocks/];
        clock-names = [aroot.soc.pcie.clock_names/];
        status = [aroot.soc.pcie.status/];
```

```
};
[/if]

[for (Sequence{1, 2, 3, 4})]
[if (aroot.soc.aips_bus.name->at(i)->notEmpty())]
[aroot.soc.aips_bus.name->at(i)/]{
        compatible = [aroot.soc.aips_bus.compatible->at(i)/];
        address-cells = <[aroot.soc.aips_bus.address_cells->at(i)/]>;
        size-cells = <[aroot.soc.aips_bus.size_cells->at(i)/]>;
        reg = <[aroot.soc.aips_bus.reg->at(i)/]>;
        ranges[aroot.soc.aips_bus.ranges->at(i)/]>;

        [if (aroot.soc.aips_bus->at(i).uart.name->notEmpty())]
        [for (Sequence{1, 2, 3, 4, 5, 6, 7, 8})]
        [if (aroot.soc.aips_bus.uart->at(i).name->notEmpty())]
        [aroot.soc.aips_bus.uart->at(i).name/]{
                compatible = [aroot.soc.aips_bus.uart->at(i).compatible/];
                reg = <[aroot.soc.aips_bus.uart->at(i).reg/]>;
                interrupts = <[aroot.soc.aips_bus.uart->at(i).interrupts/]>;
                clocks = [aroot.soc.aips_bus.uart->at(i).clocks/];
                clock-names = [aroot.soc.aips_bus.uart->at(i).clock_names/];
                dmas = [aroot.soc.aips_bus.uart->at(i).dmas/];
                dma-names = [aroot.soc.aips_bus.uart->at(i).dma_names/];
                pinctrl-names = [aroot.soc.aips_bus.uart->at(i).pinctrl_names/];
                pinctrl-0 = <[aroot.soc.aips_bus.uart->at(i).pinctrl_0/]>;
                status = [aroot.soc.aips_bus.uart->at(i).status/];
        };
        [/if]
        [/for]
        [/if]

        [if (aroot.soc.aips_bus->at(i).gpio.name->notEmpty())]
        [for (Sequence{1, 2, 3, 4, 5, 6, 7, 8})]
        [if (aroot.soc.aips_bus.gpio->at(i).name->notEmpty())]
        [aroot.soc.aips_bus.gpio->at(i).name/]{
                compatible = [aroot.soc.aips_bus.gpio->at(i).compatible/];
                reg = <[aroot.soc.aips_bus.gpio->at(i).reg/]>;
                interrupts = [aroot.soc.aips_bus.gpio->at(i).interrupts/];
                gpio-controller[aroot.soc.aips_bus.gpio->at(i).gpio_controller/];
                gpio-cells = <[aroot.soc.aips_bus.gpio->at(i).gpio_cells/]>;
                interrupt-controller[aroot.soc.aips_bus.gpio->at(i).interrupt_controller/];
                interrupt-cells = <[aroot.soc.aips_bus.gpio->at(i).interrupt_cells/]>;
        };
        [/if]
        [/for]
        [/if]

        [if (aroot.soc.aips_bus->at(i).i2c.name->notEmpty())]
        [for (Sequence{1, 2, 3, 4, 5, 6, 7, 8})]
        [if (aroot.soc.aips_bus.i2c->at(i).name->notEmpty())]
        [aroot.soc.aips_bus.i2c->at(i).name/]{
                compatible = [aroot.soc.aips_bus.i2c->at(i).compatible/];
                reg = <[aroot.soc.aips_bus.i2c->at(i).reg/]>;
                interrupts = <[aroot.soc.aips_bus.i2c->at(i).interrupts/]>;
                clocks = [aroot.soc.aips_bus.i2c->at(i).clocks/];
                status = [aroot.soc.aips_bus.i2c->at(i).status/];
        };
        [/if]
        [/for]
        [/if]

        [if (aroot.soc.aips_bus->at(i).pwm.name->notEmpty())]
        [for (Sequence{1, 2, 3, 4, 5, 6, 7, 8})]
        [if (aroot.soc.aips_bus.pwm->at(i).name->notEmpty())]
        [aroot.soc.aips_bus.pwm->at(i).name/]{
                compatible = [aroot.soc.aips_bus.pwm->at(i).compatible/];
                pwm-cells = <[aroot.soc.aips_bus.pwm->at(i).pwm_cells/]>;
                reg = <[aroot.soc.aips_bus.pwm->at(i).reg/]>;
                interrupts = <[aroot.soc.aips_bus.pwm->at(i).interrupts/]>;
                clocks = [aroot.soc.aips_bus.pwm->at(i).clocks/];
                clock-names = [aroot.soc.aips_bus.pwm->at(i).clock_names/];
        };
        [/if]
        [/for]
        [/if]
```

```
[if (aroot.soc.aips_bus->at(i).usb.name->notEmpty())]
[for (Sequence{1, 2, 3, 4, 5, 6, 7, 8})]
[if (aroot.soc.aips_bus.usb->at(i).name->notEmpty())]
[aroot.soc.aips_bus.usb->at(i).name/]{
		compatible = [aroot.soc.aips_bus.usb->at(i).compatible/];
		reg = <[aroot.soc.aips_bus.usb->at(i).reg/]>;
		interrupts = <[aroot.soc.aips_bus.usb->at(i).interrupts/]>;
		clocks = [aroot.soc.aips_bus.usb->at(i).clocks/];
		status = [aroot.soc.aips_bus.usb->at(i).status/];
};
[/if]
[/for]
[/if]

[if (aroot.soc.aips_bus->at(i).flexcan.name->notEmpty())]
[for (Sequence{1, 2, 3, 4})]
[if (aroot.soc.aips_bus.flexcan->at(i).name->notEmpty())]
[aroot.soc.aips_bus.flexcan->at(i).name/]{
		compatible = [aroot.soc.aips_bus.flexcan->at(i).compatible/];
		reg = <[aroot.soc.aips_bus.flexcan->at(i).reg/]>;
		interrupts = <[aroot.soc.aips_bus.flexcan->at(i).interrupts/]>;
		clocks = [aroot.soc.aips_bus.flexcan->at(i).clocks/];
		clock-names = [aroot.soc.aips_bus.flexcan->at(i).clock_names/];
		stop-mode = <[aroot.soc.aips_bus.flexcan->at(i).stop_mode/]>;
		status = [aroot.soc.aips_bus.flexcan->at(i).status/];
};
[/if]
[/for]
[/if]

[if (aroot.soc.aips_bus->at(i).caam.name->notEmpty())]
[aroot.soc.aips_bus->at(i).caam.name/]{
		compatible = [aroot.soc.aips_bus->at(i).caam.compatible/];
		address-cells = <[aroot.soc.aips_bus->at(i).caam.address_cells/]>;
		size-cells = <[aroot.soc.aips_bus->at(i).caam.size_cells/]>;
		reg = <[aroot.soc.aips_bus->at(i).caam.reg/]>;
		ranges = <[aroot.soc.aips_bus->at(i).caam.ranges/]>;
		interrupt-parent = <[aroot.soc.aips_bus->at(i).caam.interrupt_parent/]>;
		clocks = [aroot.soc.aips_bus->at(i).caam.clocks/];
		clock-names = [aroot.soc.aips_bus->at(i).caam.clock_names/];

		[if (aroot.soc.aips_bus->at(i).caam.jr.name->notEmpty())]
		[for (Sequence{1, 2, 3, 4})]
		[if (aroot.soc.aips_bus.caam.jr->at(i).name->notEmpty())]
		[aroot.soc.aips_bus.caam.jr->at(i).name/]{
			compatible = [aroot.soc.aips_bus.caam.jr->at(i).compatible/];
			reg = <[aroot.soc.aips_bus.caam.jr->at(i).reg/]>;
			interrupt-parent = <[aroot.soc.aips_bus.caam.jr->at(i).interrupt_parent/]>;
			interrupts = <[aroot.soc.aips_bus.caam.jr->at(i).interrupts/]>;
		};
		[/if]
		[/for]
		[/if]

};
[/if]

[if (aroot.soc.aips_bus->at(i).usbphy.name->notEmpty())]
[for (Sequence{1, 2, 3, 4})]
[if (aroot.soc.aips_bus.usbphy->at(i).name->notEmpty())]
[aroot.soc.aips_bus.usbphy->at(i).name/]{
		compatible = [aroot.soc.aips_bus.usbphy->at(i).compatible/];
		reg = <[aroot.soc.aips_bus.usbphy->at(i).reg/]>;
		interrupts = <[aroot.soc.aips_bus.usbphy->at(i).interrupts/]>;
		clocks = [aroot.soc.aips_bus.usbphy->at(i).clocks/];
		phy-3p0-supply = <[aroot.soc.aips_bus.usbphy->at(i).phy_3p0_supply/]>;
		fsl,anatop = <[aroot.soc.aips_bus.usbphy->at(i).anatop/]>;
};
[/if]
[/for]
[/if]

[if (aroot.soc.aips_bus->at(i).clks.name->notEmpty())]
[aroot.soc.aips_bus->at(i).clks.name/]{
		compatible = [aroot.soc.aips_bus->at(i).clks.compatible/];
		reg = <[aroot.soc.aips_bus->at(i).clks.reg/]>;
```

```
                interrupts = [aroot.soc.aips_bus->at(i).clks.interrupts/];
                clock-cells = <[aroot.soc.aips_bus->at(i).clks.clock_cells/]>;
        };
        [/if]

        [if (aroot.soc.aips_bus->at(i).wdog.name->notEmpty())]
        [for (Sequence{1, 2, 3, 4})]
        [if (aroot.soc.aips_bus.wdog->at(i).name->notEmpty())]
        [aroot.soc.aips_bus.wdog->at(i).name/]{
                compatible = [aroot.soc.aips_bus.wdog->at(i).compatible/];
                reg = <[aroot.soc.aips_bus.wdog->at(i).reg/]>;
                interrupts = <[aroot.soc.aips_bus.wdog->at(i).interrupts/]>;
                clocks = [aroot.soc.aips_bus.wdog->at(i).clocks/];
                status = [aroot.soc.aips_bus.wdog->at(i).status/];
        };
        [/if]
        [/for]
        [/if]

        [if (aroot.soc.aips_bus->at(i).fec.name->notEmpty())]
        [aroot.soc.aips_bus->at(i).fec.name/]{
                compatible = [aroot.soc.aips_bus->at(i).fec.compatible/];
                reg = <[aroot.soc.aips_bus->at(i).fec.reg/]>;
                interrupts-extended = [aroot.soc.aips_bus->at(i).fec.interrupts_extended/];
                clocks = [aroot.soc.aips_bus->at(i).fec.clocks/];
                clock-names = [aroot.soc.aips_bus->at(i).fec.clock_names/];
                status = [aroot.soc.aips_bus->at(i).fec.status/];
        };
        [/if]

        [if (aroot.soc.aips_bus->at(i).iomuxc.name->notEmpty())]
        [aroot.soc.aips_bus->at(i).iomuxc.name/]{
                compatible = [aroot.soc.aips_bus->at(i).iomuxc.compatible/];
                reg = <[aroot.soc.aips_bus->at(i).iomuxc.reg/]>;
                pinctrl-names = <[aroot.soc.aips_bus->at(i).iomuxc.pinctrl_names/]>;
                pinctrl-0 = <[aroot.soc.aips_bus->at(i).iomuxc.pinctrl_0/]>;

                [if (aroot.soc.aips_bus->at(i).iomuxc.platform_gpios.name->notEmpty())]
                [aroot.soc.aips_bus->at(i).iomuxc.platform_gpios.name/]{
                        [if (aroot.soc.aips_bus->at(i).iomuxc.platform_gpios.gpio_pin_grp.name->notEmpty())]
                        [aroot.soc.aips_bus->at(i).iomuxc.platform_gpios.gpio_pin_grp.name/]{
                                fsl,pins = <[aroot.soc.aips_bus->at(i).iomuxc.platform_gpios.gpio_pin_grp.pins/]>;
                        };
                        [/if]
                };
                [/if]
        [/if]

        [if (aroot.soc.aips_bus->at(i).ldb.name->notEmpty())]
        [aroot.soc.aips_bus->at(i).ldb.name/]{
                address-cells = <[aroot.soc.aips_bus->at(i).ldb.address_cells/]>;
                size-cells = <[aroot.soc.aips_bus->at(i).ldb.size_cells/]>;
                gpr = <[aroot.soc.aips_bus->at(i).ldb.size_cells/]>;
                compatible = <[aroot.soc.aips_bus->at(i).ldb.gpr/]>;
                clocks = [aroot.soc.aips_bus->at(i).ldb.clocks/];
                clock-names = [aroot.soc.aips_bus->at(i).ldb.clock_names/];
                status = [aroot.soc.aips_bus->at(i).ldb.status/];

                [if (aroot.soc.aips_bus->at(i).ldb.lvds_channel.name->notEmpty())]
                [for (Sequence{1, 2, 3, 4})]
                [if (aroot.soc.aips_bus.ldb.lvds_channel->at(i).name->notEmpty())]
                [aroot.soc.aips_bus.ldb.lvds_channel->at(i).name/]{
                        reg = <[aroot.soc.aips_bus.ldb.lvds_channel->at(i).reg/]>;
                        fsl,data-mapping = [aroot.soc.aips_bus.ldb.lvds_channel->at(i).data_mapping/];
                        fsl,data-width = <[aroot.soc.aips_bus.ldb.lvds_channel->at(i).data_width/]>;
                        primary[aroot.soc.aips_bus.ldb.lvds_channel->at(i).primary/];
                        status = [aroot.soc.aips_bus.ldb.lvds_channel->at(i).status/];

                        [aroot.soc.aips_bus.ldb.lvds_channel->at(i).display_timings.name/]{
                                native-mode = <[aroot.soc.aips_bus.ldb.lvds_channel->at(i).display_timings.native_mode/]>;
                                [aroot.soc.aips_bus.ldb.lvds_channel->at(i).display_timings.timing.name/]{
```

```
                                                clock-frequency =
<[aroot.soc.aips_bus.ldb.lvds_channel->at(i).display_timings.timing.clock_frequency/]>;
                                                hactive = <[aroot.soc.aips_bus.ldb.lvds_channel-
>at(i).display_timings.timing.hactive/]>;
                                                vactive = <[aroot.soc.aips_bus.ldb.lvds_channel-
>at(i).display_timings.timing.vactive/]>;
                                                hback-porch = <[aroot.soc.aips_bus.ldb.lvds_channel-
>at(i).display_timings.timing.hback_porch/]>;
                                                hfront-porch = <[aroot.soc.aips_bus.ldb.lvds_channel-
>at(i).display_timings.timing.hfront_porch/]>;
                                                vback-porch = <[aroot.soc.aips_bus.ldb.lvds_channel-
>at(i).display_timings.timing.vback_porch/]>;
                                                vfront-porch = <[aroot.soc.aips_bus.ldb.lvds_channel-
>at(i).display_timings.timing.vfront_porch/]>;
                                                hsync-len = <[aroot.soc.aips_bus.ldb.lvds_channel-
>at(i).display_timings.timing.hsync_len/]>;
                                                vsync-len = <[aroot.soc.aips_bus.ldb.lvds_channel-
>at(i).display_timings.timing.vsync_len/]>;
                                                hsync-active = <[aroot.soc.aips_bus.ldb.lvds_channel-
>at(i).display_timings.timing.hsync_active/]>;
                                                vsync-active = <[aroot.soc.aips_bus.ldb.lvds_channel-
>at(i).display_timings.timing.vsync_active/]>;
                                        };
                                };
                        };
                        [/if]
                        [/for]
                        [/if]

                };
                [/if]

                [if (aroot.soc.aips_bus->at(i).spba_bus.name->notEmpty())]
                [aroot.soc.aips_bus->at(i).spba_bus.name/]{
                        compatible = [aroot.soc.aips_bus->at(i).spba_bus.compatible/];
                        address-cells = <[aroot.soc.aips_bus->at(i).spba_bus.address_cells/]>;
                        size-cells = <[aroot.soc.aips_bus->at(i).spba_bus.size_cells/]>;
                        reg = <[aroot.soc.aips_bus->at(i).spba_bus.reg/]>;
                        ranges[aroot.soc.aips_bus->at(i).spba_bus.ranges/];

                        [if (aroot.soc.aips_bus->at(i).spba_bus.ssi.name->notEmpty())]
                        [for (Sequence{1, 2, 3, 4})]
                        [if (aroot.soc.aips_bus.spba_bus.ssi->at(i).name->notEmpty())]
                        [aroot.soc.aips_bus.spba_bus.ssi->at(i).name/]{
                                compatible = [aroot.soc.aips_bus.spba_bus.ssi->at(i).compatible/];
                                reg = <[aroot.soc.aips_bus.spba_bus.ssi->at(i).reg/]>;
                                interrupts = <[aroot.soc.aips_bus.spba_bus.ssi->at(i).interrupts/]>;
                                clocks = [aroot.soc.aips_bus.spba_bus.ssi->at(i).clocks/];
                                clock-names = [aroot.soc.aips_bus.spba_bus.ssi->at(i).clock_names/];
                                dmas = [aroot.soc.aips_bus.spba_bus.ssi->at(i).dmas/];
                                dma-names = [aroot.soc.aips_bus.spba_bus.ssi->at(i).dma_names/];
                                fsl,fifo-depth = <[aroot.soc.aips_bus.spba_bus.ssi->at(i).fifo_depth/]>;
                                fsl,ssi-dma-events = <[aroot.soc.aips_bus.spba_bus.ssi-
>at(i).ssi_dma_events/]>;
                                status = [aroot.soc.aips_bus.spba_bus.ssi->at(i).status/];
                        };
                        [/if]
                        [/for]
                        [/if]

                        [if (aroot.soc.aips_bus->at(i).spba_bus.ecspi.name->notEmpty())]
                        [for (Sequence{1, 2, 3, 4})]
                        [if (aroot.soc.aips_bus.spba_bus.ecspi->at(i).name->notEmpty())]
                        [aroot.soc.aips_bus.spba_bus.ecspi->at(i).name/]{
                                address-cells = <[aroot.soc.aips_bus.spba_bus.ecspi->at(i).address_cells/]>;
                                size-cells = <[aroot.soc.aips_bus.spba_bus.ecspi->at(i).size_cells/]>;
                                compatible = [aroot.soc.aips_bus.spba_bus.ecspi->at(i).compatible/];
                                reg = <[aroot.soc.aips_bus.spba_bus.ecspi->at(i).reg/]>;
                                interrupts = <[aroot.soc.aips_bus.spba_bus.ecspi->at(i).interrupts/]>;
                                clocks = [aroot.soc.aips_bus.spba_bus.ecspi->at(i).clocks/];
                                clock-names = [aroot.soc.aips_bus.spba_bus.ecspi->at(i).clock_names/];
                                dmas = [aroot.soc.aips_bus.spba_bus.ecspi->at(i).dmas/];
                                dma-names = [aroot.soc.aips_bus.spba_bus.ecspi->at(i).dma_names/];
                                fsl,spi-num-chipselects = <[aroot.soc.aips_bus.spba_bus.ecspi-
>at(i).spi_num_chipselects/]>;
                                cs-gpios = [aroot.soc.aips_bus.spba_bus.ecspi->at(i).cs_gpios/];
```

```
                                        pinctrl-names = [aroot.soc.aips_bus.spba_bus.ecspi->at(i).pinctrl_names/];
                                        pinctrl-0 = [aroot.soc.aips_bus.spba_bus.ecspi->at(i).pinctrl_0/];
                                        status = [aroot.soc.aips_bus.spba_bus.ecspi->at(i).status/];
                                };
                                [/if]
                                [/for]
                                [/if]

                                [if (aroot.soc.aips_bus->at(i).spba_bus.spdif.name->notEmpty())]
                                [for (Sequence{1, 2})]
                                [if (aroot.soc.aips_bus.spba_bus.spdif->at(i).name->notEmpty())]
                                [aroot.soc.aips_bus.spba_bus.spdif->at(i).name/]{
                                        compatible = [aroot.soc.aips_bus.spba_bus.spdif->at(i).compatible/];
                                        reg = <[aroot.soc.aips_bus.spba_bus.spdif->at(i).reg/]>;
                                        interrupts = <[aroot.soc.aips_bus.spba_bus.spdif->at(i).interrupts/]>;
                                        dmas = [aroot.soc.aips_bus.spba_bus.spdif->at(i).dmas/];
                                        dma-names = [aroot.soc.aips_bus.spba_bus.spdif->at(i).dma_names/];
                                        clocks = [aroot.soc.aips_bus.spba_bus.spdif->at(i).clocks/];
                                        clock-names = [aroot.soc.aips_bus.spba_bus.spdif->at(i).clock_names/];
                                        status = [aroot.soc.aips_bus.spba_bus.spdif->at(i).status/];
                                };
                                [/if]
                                [/for]
                                [/if]

                                [if (aroot.soc.aips_bus->at(i).spba_bus.esai.name->notEmpty())]
                                [for (Sequence{1, 2})]
                                [if (aroot.soc.aips_bus.spba_bus.esai->at(i).name->notEmpty())]
                                [aroot.soc.aips_bus.spba_bus.esai->at(i).name/]{
                                        compatible = [aroot.soc.aips_bus.spba_bus.esai->at(i).compatible/];
                                        reg = <[aroot.soc.aips_bus.spba_bus.esai->at(i).reg/]>;
                                        interrupts = <[aroot.soc.aips_bus.spba_bus.esai->at(i).interrupts/]>;
                                        clocks = [aroot.soc.aips_bus.spba_bus.esai->at(i).clocks/];
                                        clock-names = [aroot.soc.aips_bus.spba_bus.esai->at(i).clock_names/];
                                        dmas = [aroot.soc.aips_bus.spba_bus.esai->at(i).dmas/];
                                        dma-names = [aroot.soc.aips_bus.spba_bus.esai->at(i).dma_names/];
                                        status = [aroot.soc.aips_bus.spba_bus.esai->at(i).status/];
                                };
                                [/if]
                                [/for]
                                [/if]

                        };
                        [/if]
                };

                [/if]
                [/for]
            };
 };
[/file]
[/template]
```

Ek 2 Üretilen DT Kodu

```
/*
 *Copyright 2018 Kent Kart Ege Elektronik A.S.
 *
 *This code contains iMX6dl based device tree structure for driver computer board
 *
 */

#include <dt-bindings/interrupt-controller/irq.h>
#include "imx6dl-pinfunc.h"
#include "imx6qdl.dtsi"
#include <dt-bindings/clock/imx6qdl-clock.h>
#include <dt-bindings/gpio/gpio.h>
#include <dt-bindings/interrupt-controller/arm-gic.h>
#include "skeleton.dtsi"
#include <dt-bindings/input/input.h>

/{
	model = Kentkart i.MX6 DualLite Smart Device Board;
	compatible = "fsl,imx6dl-sabresd", "fsl,imx6dl";

	cpus{
	  address-cells = <1>;
	  size-cells = <0>;

	  cpu0{
			compatible = "arm,cortex-a9";
			device_type = "cpu";
			reg = <0>;
			next-level-cache = <&L2>;
			operating-points = <
				996000 1275000
				792000 1175000
				396000 1150000
				>;
			fsl,soc-operating-points = <
				996000 1175000
				792000 1175000
				396000 1175000
				>;
			clock-latency = <61036>;
			clocks = <&clks IMX6QDL_CLK_ARM>,
				<&clks IMX6QDL_CLK_PLL2_PFD2_396M>,
				<&clks IMX6QDL_CLK_STEP>,
				<&clks IMX6QDL_CLK_PLL1_SW>,
				<&clks IMX6QDL_CLK_PLL1_SYS>,
				<&clks IMX6QDL_PLL1_BYPASS>,
				<&clks IMX6QDL_CLK_PLL1>,
				<&clks IMX6QDL_PLL1_BYPASS_SRC>;
			clock-names = "arm", "pll2_pfd2_396m", "step",
					"pll1_sw", "pll1_sys", "pll1_bypass", "pll1", "pll1_bypass_src";
			};

	  cpu1{
			compatible = "arm,cortex-a9";
			device_type = "cpu";
			reg = <1>;
			next-level-cache = <&L2>;
			};
	};

	memory{
	  reg = <0x10000000 0x40000000>;
	};

	aliases{
	  gpio_s = &gpio1;
	  i2c_s = &i2c1;
	  mmc_s = &usdhc1;
	  serial_s = &uart1;
	  spi_s = &ecspi1;
	  usbphy_s = &usbphy1;
	  mxcfb_s = &mxcfb1;
```

```
};

gpio-keys{
   compatible = "gpio-keys";
   pinctrl-names = "default";
   pinctrl-0 = <&pinctrl_gpio_keys>;

   power{
                label = "Power Button";
                gpios = <&gpio2 06 1>;
                gpio-key,wakeup;
                linux,code = <KEY_POWER>;
   };
};

sound{
   compatible = "fsl,imx-audio-cs42l73";
   model = "cs42l73";
   cpu-dai = <&ssi1>;
   audio-codec = <&codec>;
   audio-routing = "MIC_IN", "Mic Jack",
                "Mic Jack", "Mic Bias",
                "Headphone Jack","HP_OUT";
   mux-int-port = <1>;
   mux-ext-port = <3>;
};

sound-spdif{
   compatible = "fsl,imx-audio-spdif";
   model = "imx-spdif";
   spdif-controller = <&spdif>;
   spdif-in;
   spdif-out;
};

sound-hdmi{
   compatible = "fsl,imx6q-audio-hdmi","fsl,imx-audio-hdmi";
   model = "imx-audio-hdmi";
   spdif-controller = <&hdmi_audio>;
};

backlight{
   compatible = "pwm-backlight";
   pwms = <&pwm2 0 5000000>;
   brightness-levels = <
                0 1 2 3 4 5 6 7 8 9
                >;
   default-brightness-level = <255>;
   status = "okay";
};

battery{
   offset-charger = 1485;
   offset-discharger = <1464>;
   offset-usb-charger = <1285>;
};

lcd{
   compatible = "fsl,lcd";
   ipu_id = <0>;
   disp_id = <0>;
   default_ifmt = "RGB24";
   pinctrl-names = "default";
   pinctrl-0 = <&pinctrl_ipu1>;
   status = "okay";
};

intc{
   compatible = "arm,cortex-a9-gic";
   interrupt-cells = <3>;
   address-cells = <1>;
   size-cells = <1>;
   interrupt-controller;
   reg = <0x00a01000 0x1000>, <0x00a00100 0x100>;
};
```

```
soc{
    address-cells = <1>;
    size-cells = <1>;
    compatible = "simple-bus";
    interrupt-parent = <&intc>;
    ranges;

    gpmi-nand{
                compatible = "fsl,imx6q-gpmi-nand";
                address-cells = <1>;
                size-cells = <1>;
                reg = <0x00112000 0x2000>, <0x00114000 0x2000>;
                reg-names = "gpmi-nand", "bch";
                interrupts = <0 15 IRQ_TYPE_LEVEL_HIGH>;
                interrupt-names = "bch";
                clocks = <&clks IMX6QDL_CLK_GPMI_IO>,
                        <&clks IMX6QDL_CLK_GPMI_APB>,
                        <&clks IMX6QDL_CLK_GPMI_BCH>,
                        <&clks IMX6QDL_CLK_GPMI_BCH_APB>,
                        <&clks IMX6QDL_CLK_PER1_BCH> ;
                clock-names = "gpmi_io", "gpmi_apb", "gpmi_bch",
                                "gpmi_bch_apb", "per1_bch";
                dmas = <&dma_apbh 0>;
                dma-names = "rx-tx";
                status = "disabled";
    };

    ipu1{
                compatible = "fsl,imx6q-ipu";
                reg = <0x02400000 0x400000>;
                interrupts = <0 6 IRQ_TYPE_LEVEL_HIGH>,
                                <0 5 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_IPU1>,
                        <&clks IMX6QDL_CLK_IPU1_DI0>,
                        <&clks IMX6QDL_CLK_IPU1_DI1>,
                        <&clks IMX6QDL_CLK_IPU1_DI0_SEL>,
                        <&clks IMX6QDL_CLK_IPU1_DI1_SEL>,
                        <&clks IMX6QDL_CLK_LDB_DI0>,
                        <&clks IMX6QDL_CLK_LDB_DI1>;
                clock-names = "bus",
                                                "di0", "di1",
                                                "di0_sel", "di1_sel",
                                                "ldb_di0", "ldb_di1";
                resets = <&src 2>;
                bypass_reset = <0>;
    };

    hdmi_core{
                compatible = "fsl,imx6q-hdmi-core";
                reg = <0x00120000 0x9000>;
                clocks = <&clks IMX6QDL_CLK_HDMI_ISFR>,
                                <&clks IMX6QDL_CLK_HDMI_IAHB>,
                                <&clks IMX6QDL_CLK_HSI_TX>;
                clock-names = "hdmi_isfr", "hdmi_iahb", "mipi_core";
                status = "disabled";
    };

    hdmi_video{
                compatible = "fsl,imx6q-hdmi-video";
                reg = <0x020e0000 0x1000>;
                reg-names = "hdmi_gpr";
                interrupts = <0 115 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_HDMI_ISFR>,
                                <&clks IMX6QDL_CLK_HDMI_IAHB>,
                                <&clks IMX6QDL_CLK_HSI_TX>;
                clock-names = "hdmi_isfr", "hdmi_iahb", "mipi_core";
                status = "disabled";
    };

    hdmi_audio{
                compatible = "fsl,imx6q-hdmi-audio";
                clocks = <&clks IMX6QDL_CLK_HDMI_ISFR>,
                                <&clks IMX6QDL_CLK_HDMI_IAHB>,
                                <&clks IMX6QDL_CLK_HSI_TX>;
                clock-names = "hdmi_isfr", "hdmi_iahb", "mipi_core";
                dmas = <&sdma 2 24 0>;
```

```
                dma-names = "tx";
                status = "disabled";
        };

        hdmi_cec{
                compatible = "fsl,imx6q-hdmi-cec";
                interrupts = <0 115 IRQ_TYPE_LEVEL_HIGH>;
                status = "disabled";
        };

        gpu{
                compatible = "fsl,imx6dl-gpu", "fsl,imx6q-gpu";
                reg = <0x00130000 0x4000>, <0x00134000 0x4000>,
                                <0x0 0x0>;
                reg-names = "iobase_3d", "iobase_2d",
                                "phys_baseaddr";
                interrupts = <0 9 IRQ_TYPE_LEVEL_HIGH>,
                                <0 10 IRQ_TYPE_LEVEL_HIGH>;
                interrupt-names = "irq_3d", "irq_2d";
                clocks = <&clks IMX6QDL_CLK_OPENVG_AXI>, <&clks IMX6QDL_CLK_GPU3D_AXI>,
                        <&clks IMX6QDL_CLK_GPU2D_CORE>, <&clks IMX6QDL_CLK_GPU3D_CORE>,
                        <&clks IMX6QDL_CLK_DUMMY>;
                clock-names = "gpu2d_axi_clk", "gpu3d_axi_clk",
                                                "gpu2d_clk", "gpu3d_clk",
                                                "gpu3d_shader_clk";
                resets = <&src 0>, <&src 3>;
                reset-names = "gpu3d", "gpu2d";
                power-domains = &gpc 1;
        };

        l2_cache{
                compatible = "arm,pl310-cache";
                regs = <0x00a02000 0x1000>;
                interrupts = <0 92 IRQ_TYPE_LEVEL_HIGH>;
                cache-unified;
                cache-level = <2>;
                arm,tag-latecy = <4 2 3>;
                arm,data-latecy = <4 2 3>;
        };

        timer{
                compatible = "arm,cortex-a9-twd-timer";
                reg = <0x00a00600 0x20>;
                interrupts = <1 13 0xf01>;
                clocks = <&clks IMX6QDL_CLK_TWD>;
        };

        pcie{
                compatible = "fsl,imx6q-pcie", "snps,dw-pcie";
                reg = <0x01ffc000 0x4000>, <0x01f00000 0x80000>;
                reg-names = "dbi", "config";
                address-cells = <3>;
                size-cells = <2>;
                device_type = "pci";
                ranges = 0x81000000 0 0  0x01f80000 0 0x00010000
                                0x82000000 0 0x01000000 0x01000000 0 0x00f00000;
                num-lanes = <1>;
                interrupts = <GIC_SPI 123 IRQ_TYPE_LEVEL_HIGH>,
                                <GIC_SPI 120 IRQ_TYPE_LEVEL_HIGH>;
                interrupt-names = "inta", "msi";
                interrupt-cells = <1>;
                interrupt-map-mask = <0 0 0 0x7>;
                interrupt-map = <0 0 0 1 &intc 0 123 IRQ_TYPE_LEVEL_HIGH>,
                                                        <0 0 0 2 &intc 0 122 IRQ_TYPE_LEVEL_HIGH>,
                                                        <0 0 0 3 &intc 0 121 IRQ_TYPE_LEVEL_HIGH>,
                                                        <0 0 0 4 &intc 0 120 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_PCIE_REF_125M>,
                                <&clks IMX6QDL_CLK_SATA_REF_100M>,
                                <&clks IMX6QDL_CLK_LVDS1_GATE>, <&clks
IMX6QDL_CLK_PCIE_AXI>;
                clock-names = "pcie_phy", "ref_100m", "pcie_bus", "pcie";
                status = "disabled";
        };

        aips1{
                compatible = "fsl,aips-bus", "simple-bus";
```

```
address-cells = <1>;
size-cells = <1>;
reg = <0x02000000 0x100000>;
ranges>;

gpio1{
        compatible = "fsl,imx6q-gpio", "fsl,imx35-gpio";
        reg = <0x0209c000 0x4000>;
        interrupts = <0 66 IRQ_TYPE_LEVEL_HIGH>,
                     <0 67 IRQ_TYPE_LEVEL_HIGH>;
        gpio-controller;
        gpio-cells = <2>;
        interrupt-controller;
        interrupt-cells = <2>;
};

gpio2{
        compatible = "fsl,imx6q-gpio", "fsl,imx35-gpio";
        reg = <0x020a0000 0x4000>;
        interrupts = <0 68 IRQ_TYPE_LEVEL_HIGH>,
                     <0 69 IRQ_TYPE_LEVEL_HIGH>;
        gpio-controller;
        gpio-cells = <2>;
        interrupt-controller;
        interrupt-cells = <2>;
};

gpio3{
        compatible = "fsl,imx6q-gpio", "fsl,imx35-gpio";
        reg = <0x020a4000 0x4000>;
        interrupts = <0 70 IRQ_TYPE_LEVEL_HIGH>,
                     <0 71 IRQ_TYPE_LEVEL_HIGH>;
        gpio-controller;
        gpio-cells = <2>;
        interrupt-controller;
        interrupt-cells = <2>;
};

gpio4{
        compatible = "fsl,imx6q-gpio", "fsl,imx35-gpio";
        reg = <0x020a8000 0x4000>;
        interrupts = <0 72 IRQ_TYPE_LEVEL_HIGH>,
                     <0 73 IRQ_TYPE_LEVEL_HIGH>;
        gpio-controller;
        gpio-cells = <2>;
        interrupt-controller;
        interrupt-cells = <2>;
};

gpio5{
        compatible = "fsl,imx6q-gpio", "fsl,imx35-gpio";
        reg = <0x020ac000 0x4000>;
        interrupts = <0 74 IRQ_TYPE_LEVEL_HIGH>,
                     <0 75 IRQ_TYPE_LEVEL_HIGH>;
        gpio-controller;
        gpio-cells = <2>;
        interrupt-controller;
        interrupt-cells = <2>;
};

gpio6{
        compatible = "fsl,imx6q-gpio", "fsl,imx35-gpio";
        reg = <0x020b0000 0x4000>;
        interrupts = <0 76 IRQ_TYPE_LEVEL_HIGH>,
                     <0 77 IRQ_TYPE_LEVEL_HIGH>;
        gpio-controller;
        gpio-cells = <2>;
        interrupt-controller;
        interrupt-cells = <2>;
};

gpio7{
        compatible = "fsl,imx6q-gpio", "fsl,imx35-gpio";
        reg = <0x020b4000 0x4000>;
        interrupts = <0 78 IRQ_TYPE_LEVEL_HIGH>,
                     <0 79 IRQ_TYPE_LEVEL_HIGH>;
```

```
                gpio-controller;
                gpio-cells = <2>;
                interrupt-controller;
                interrupt-cells = <2>;
};

pwm1{
                compatible = "fsl,imx6q-pwm", "fsl,imx27-pwm";
                pwm-cells = <2>;
                reg = <0x02080000 0x4000>;
                interrupts = <0 83 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_IPG>,
                                <&clks IMX6QDL_CLK_PWM1>;
                clock-names = "ipg", "per";
};

pwm2{
                compatible = "fsl,imx6q-pwm", "fsl,imx27-pwm";
                pwm-cells = <2>;
                reg = <0x02084000 0x4000>;
                interrupts = <0 84 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_IPG>,
                                <&clks IMX6QDL_CLK_PWM2>;
                clock-names = "ipg", "per";
};

pwm3{
                compatible = "fsl,imx6q-pwm", "fsl,imx27-pwm";
                pwm-cells = <2>;
                reg = <0x02088000 0x4000>;
                interrupts = <0 85 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_IPG>,
                                <&clks IMX6QDL_CLK_PWM3>;
                clock-names = "ipg", "per";
};

pwm4{
                compatible = "fsl,imx6q-pwm", "fsl,imx27-pwm";
                pwm-cells = <2>;
                reg = <0x0208c000 0x4000>;
                interrupts = <0 86 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_IPG>,
                                <&clks IMX6QDL_CLK_PWM4>;
                clock-names = "ipg", "per";
};

flexcan1{
                compatible = "fsl,imx6q-flexcan";
                reg = <0x02090000 0x4000>;
                interrupts = <0 110 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_CAN1_IPG>,
                                <&clks IMX6QDL_CLK_CAN1_SERIAL>;
                clock-names = "ipg", "per";
                stop-mode = <&gpr 0x34 28 0x10 17>;
                status = "disabled";
};

flexcan2{
                compatible = "fsl,imx6q-flexcan";
                reg = <0x02094000 0x4000>;
                interrupts = <0 111 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_CAN2_IPG>,
                                <&clks IMX6QDL_CLK_CAN2_SERIAL>;
                clock-names = "ipg", "per";
                stop-mode = <&gpr 0x34 29 0x10 18>;
                status = "disabled";
};

usbphy1{
                compatible = "fsl,imx6q-usbphy", "fsl,imx23-usbphy";
                reg = <0x020c9000 0x1000>;
                interrupts = <0 44 IRQ_TYPE_LEVEL_HIGH>;
                clocks = &clks IMX6QDL_CLK_USBPHY1;
                phy-3p0-supply = <&reg_3p0>;
                fsl,anatop = <&anatop>;
};
```

```
                usbphy2{
                        compatible = "fsl,imx6q-usbphy", "fsl,imx23-usbphy";
                        reg = <0x020ca000 0x1000>;
                        interrupts = <0 45 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = &clks IMX6QDL_CLK_USBPHY2;
                        phy-3p0-supply = <&reg_3p0>;
                        fsl,anatop = <&anatop>;
                };

                clks{
                        compatible = "fsl,imx6q-ccm";
                        reg = <0x020c4000 0x4000>;
                        interrupts = <0 87 IRQ_TYPE_LEVEL_HIGH>,
                                <0 88 IRQ_TYPE_LEVEL_HIGH>;
                        clock-cells = <1>;
                };

                wdog1{
                        compatible = "fsl,imx6q-wdt", "fsl,imx21-wdt";
                        reg = <0x020bc000 0x4000>;
                        interrupts = <0 80 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = &clks IMX6QDL_CLK_DUMMY;
                        status = "okay";
                };

                wdog2{
                        compatible = "fsl,imx6q-wdt", "fsl,imx21-wdt";
                        reg = <0x020c0000 0x4000>;
                        interrupts = <0 81 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = &clks IMX6QDL_CLK_DUMMY;
                        status = "disabled";
                };

                iomuxc{
                        compatible = "fsl,imx6dl-iomuxc", "fsl,imx6q-iomuxc";
                        reg = <0x020e0000 0x4000>;
                        pinctrl-names = <"default">;
                        pinctrl-0 = <&pinctrl_hog>;

                        imx6qdl-smarc{
                                pinctrl_hog{
                                        fsl,pins = <MX6QDL_PAD_GPIO_4__GPIO1_IO04
        0x80000000,

        MX6QDL_PAD_GPIO_5__GPIO1_IO05      0x80000000,

        MX6QDL_PAD_GPIO_6__GPIO1_IO06      0x80000000,

        MX6QDL_PAD_NANDF_D0__GPIO2_IO00              0x80000000,

        MX6QDL_PAD_NANDF_D2__GPIO2_IO02,

        MX6QDL_PAD_NANDF_D3__GPIO2_IO03>;
                                        };
                        };

                        ldb{
                                address-cells = <1>;
                                size-cells = <0>;
                                gpr = <0>;
                                compatible = <&gpr>;
                                clocks = <&clks IMX6QDL_CLK_LDB_DI0>, <&clks IMX6QDL_CLK_LDB_DI1>,
                                        <&clks                IMX6QDL_CLK_IPU1_DI0_SEL>,                <&clks
IMX6QDL_CLK_IPU1_DI1_SEL>,
                                        <&clks IMX6QDL_CLK_IPU2_DI0_SEL>,
                                        <&clks              IMX6QDL_CLK_LDB_DI0_DIV_3_5>,              <&clks
IMX6QDL_CLK_LDB_DI1_DIV_3_5>,
                                        <&clks                IMX6QDL_CLK_LDB_DI0_DIV_7>,              <&clks
IMX6QDL_CLK_LDB_DI1_DIV_7>,
                                        <&clks              IMX6QDL_CLK_LDB_DI0_DIV_SEL>,              <&clks
IMX6QDL_CLK_LDB_DI1_DIV_SEL>;
                                clock-names = "ldb_di0", "ldb_di1",
                                                "di0_sel", "di1_sel",
                                                "di2_sel",
                                                "ldb_di0_div_3_5", "ldb_di1_div_3_5",
```

```
                                        "ldb_di0_div_7", "ldb_di1_div_7",
                                        "ldb_di0_div_sel", "ldb_di1_div_sel";
                status = "okay";

                lvds-channel0{
                        reg = <0>;
                        fsl,data-mapping = "jeida";
                        fsl,data-width = <24>;
                        primary;
                        status = "okay";

                        display-timings{
                                native-mode = <&timing0>;
                                timing0{
                                        clock-frequency = <29500000>;
                                        hactive = <800>;
                                        vactive = <480>;
                                        hback-porch = <128>;
                                        hfront-porch = <128>;
                                        vback-porch = <20>;
                                        vfront-porch = <25>;
                                        hsync-len = <10>;
                                        vsync-len = <1>;
                                        hsync-active = <0>;
                                        vsync-active = <0>;
                                };
                        };
                };

                lvds-channel1{
                        reg = <1>;
                        fsl,data-mapping = "spwg";
                        fsl,data-width = <24>;
                        primary;
                        status = "okay";

                        display-timings{
                                native-mode = <&timing1>;
                                timing1{
                                        clock-frequency = <33300000>;
                                        hactive = <480>;
                                        vactive = <480>;
                                        hback-porch = <64>;
                                        hfront-porch = <64>;
                                        vback-porch = <12>;
                                        vfront-porch = <4>;
                                        hsync-len = <128>;
                                        vsync-len = <12>;
                                        hsync-active = <0>;
                                        vsync-active = <0>;
                                };
                        };
                };
        };

        spba_bus{
                compatible = "fsl,spba-bus", "simple-bus";
                address-cells = <1>;
                size-cells = <1>;
                reg = <0x02000000 0x40000>;
                ranges;

                ssi1{
                        compatible = "fsl,imx6q-ssi","fsl,imx21-ssi";
                        reg = <0x02028000 0x4000>;
                        interrupts = <0 46 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = <&clks IMX6QDL_CLK_SSI1_IPG>,
                                 <&clks IMX6QDL_CLK_SSI1>;
                        clock-names = "ipg", "baud";
                        dmas = <&sdma 37 22 0>,
                               <&sdma 38 22 0>;
                        dma-names = "rx", "tx";
                        fsl,fifo-depth = <15>;
                        fsl,ssi-dma-events = <38 37>;
                        status = "disabled";
                };
```

```
                ssi2{
                        compatible = "fsl,imx6q-ssi","fsl,imx21-ssi";
                        reg = <0x0202c000 0x4000>;
                        interrupts = <0 47 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = <&clks IMX6QDL_CLK_SSI2_IPG>,
                                 <&clks IMX6QDL_CLK_SSI2>;
                        clock-names = "ipg", "baud";
                        dmas = <&sdma 41 22 0>,
                               <&sdma 42 22 0>;
                        dma-names = "rx", "tx";
                        fsl,fifo-depth = <15>;
                        fsl,ssi-dma-events = <42 41>;
                        status = "disabled";
                };

                ssi3{
                        compatible = "fsl,imx6q-ssi","fsl,imx21-ssi";
                        reg = <0x02030000 0x4000>;
                        interrupts = <0 48 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = <&clks IMX6QDL_CLK_SSI3_IPG>,
                                 <&clks IMX6QDL_CLK_SSI3>;
                        clock-names = "ipg", "baud";
                        dmas = <&sdma 45 22 0>,
                               <&sdma 46 22 0>;
                        dma-names = "rx", "tx";
                        fsl,fifo-depth = <15>;
                        fsl,ssi-dma-events = <46 45>;
                        status = "disabled";
                };

                ecspi1{
                        address-cells = <1>;
                        size-cells = <0>;
                        compatible = "fsl,imx6q-ecspi", "fsl,imx51-ecspi";
                        reg = <0x02008000 0x4000>;
                        interrupts = <0 31 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = <&clks IMX6QDL_CLK_ECSPI1>,
                                 <&clks IMX6QDL_CLK_ECSPI1>;
                        clock-names = "ipg", "per";
                        dmas = <&sdma 3 7 1>, <&sdma 4 7 2>;
                        dma-names = "rx", "tx";
                        fsl,spi-num-chipselects = <2>;
                        cs-gpios = <&gpio1 25 GPIO_ACTIVE_LOW>, <&gpio2 0
GPIO_ACTIVE_LOW>;
                        pinctrl-names = "default";
                        pinctrl-0 = &pinctrl_ecspi1;
                        status = "okay";
                };

                ecspi2{
                        address-cells = <1>;
                        size-cells = <0>;
                        compatible = "fsl,imx6q-ecspi", "fsl,imx51-ecspi";
                        reg = <0x0200c000 0x4000>;
                        interrupts = <0 32 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = <&clks IMX6QDL_CLK_ECSPI2>,
                                 <&clks IMX6QDL_CLK_ECSPI2>;
                        clock-names = "ipg", "per";
                        dmas = <&sdma 5 7 1>, <&sdma 6 7 2>;
                        dma-names = "rx", "tx";
                        fsl,spi-num-chipselects = <4>;
                        cs-gpios = <&gpio2 2 GPIO_ACTIVE_LOW>,<&gpio2 7
GPIO_ACTIVE_LOW>,<&gpio3 24 GPIO_ACTIVE_LOW>,<&gpio2 4 GPIO_ACTIVE_LOW>;
                        pinctrl-names = "default";
                        pinctrl-0 = &pinctrl_ecspi2;
                        status = "okay";
                };

                ecspi3{
                        address-cells = <1>;
                        size-cells = <0>;
                        compatible = "fsl,imx6q-ecspi", "fsl,imx51-ecspi";
                        reg = <0x02010000 0x4000>;
                        interrupts = <0 33 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = <&clks IMX6QDL_CLK_ECSPI3>,
```

```
                                <&clks IMX6QDL_CLK_ECSPI3>;
                        clock-names = "ipg", "per";
                        dmas = <&sdma 7 7 1>, <&sdma 8 7 2>;
                        dma-names = "rx", "tx";
                        status = "disabled";
                };

                ecspi4{
                        address-cells = <1>;
                        size-cells = <0>;
                        compatible = "fsl,imx6q-ecspi", "fsl,imx51-ecspi";
                        reg = <0x02014000 0x4000>;
                        interrupts = <0 34 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = <&clks IMX6QDL_CLK_ECSPI4>,
                                <&clks IMX6QDL_CLK_ECSPI4>;
                        clock-names = "ipg", "per";
                        dmas = <&sdma 9 7 1>, <&sdma 10 7 2>;
                        dma-names = "rx", "tx";
                        status = "disabled";
                };

                spdif{
                        compatible = "fsl,imx35-spdif";
                        reg = <0x02004000 0x4000>;
                        interrupts = <0 52 IRQ_TYPE_LEVEL_HIGH>;
                        dmas = <&sdma 14 18 0>,<&sdma 15 18 0>;
                        dma-names = "rx", "tx";
                        clocks = <&clks IMX6QDL_CLK_SPDIF_GCLK>, <&clks
IMX6QDL_CLK_OSC>,
                                <&clks IMX6QDL_CLK_SPDIF>, <&clks
IMX6QDL_CLK_ASRC>,
                                <&clks IMX6QDL_CLK_DUMMY>, <&clks
IMX6QDL_CLK_ESAI_EXTAL>,
                                <&clks IMX6QDL_CLK_IPG>, <&clks IMX6QDL_CLK_MLB>,
                                <&clks IMX6QDL_CLK_DUMMY>, <&clks
IMX6QDL_CLK_SPBA>;
                        clock-names = "core",  "rxtx0",
                                "rxtx1", "rxtx2","rxtx3",
                                "rxtx4","rxtx5", "rxtx6",
                                "rxtx7", "dma";
                        status = "disabled";
                };

                esai{
                        compatible = "fsl,imx35-esai";
                        reg = <0x02024000 0x4000>;
                        interrupts = <0 51 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = <&clks IMX6QDL_CLK_ESAI_IPG>,
                                <&clks IMX6QDL_CLK_ESAI_MEM>,
                                <&clks IMX6QDL_CLK_ESAI_EXTAL>,
                                <&clks IMX6QDL_CLK_ESAI_IPG>,
                                <&clks IMX6QDL_CLK_SPBA>;
                        clock-names = "core", "mem", "extal", "fsys", "dma";
                        dmas = <&sdma 23 21 0>,
                                 <&sdma 24 21 0>;
                        dma-names = "rx", "tx";
                        status = "disabled";
                };

        };
};

aips2{
        compatible = "fsl,aips-bus", "simple-bus";
        address-cells = <1>;
        size-cells = <1>;
        reg = <0x02100000 0x100000>;
        ranges>;

        uart1{
                compatible = "fsl,imx6q-uart", "fsl,imx21-uart";
                reg = <0x02020000 0x4000>;
                interrupts = <0 26 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_UART_IPG>,
                        <&clks IMX6QDL_CLK_UART_SERIAL>;
                clock-names = "ipg", "per";
```

```
                dmas = <&sdma 25 4 0>, <&sdma 26 4 0>;
                dma-names = "rx", "tx";
                pinctrl-names = "default";
                pinctrl-0 = <&pinctrl_uart1>;
                status = "okay";
        };

        uart2{
                compatible = "fsl,imx6q-uart", "fsl,imx21-uart";
                reg = <0x021e8000 0x4000>;
                interrupts = <0 27 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_UART_IPG>,
                         <&clks IMX6QDL_CLK_UART_SERIAL>;
                clock-names = "ipg", "per";
                dmas = <&sdma 27 4 0>, <&sdma 28 4 0>;
                dma-names = "rx", "tx";
                pinctrl-names = "default";
                pinctrl-0 = <&pinctrl_uart2>;
                status = "okay";
        };

        uart3{
                compatible = "fsl,imx6q-uart", "fsl,imx21-uart";
                reg = <0x021ec000 0x4000>;
                interrupts = <0 28 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_UART_IPG>,
                         <&clks IMX6QDL_CLK_UART_SERIAL>;
                clock-names = "ipg", "per";
                dmas = <&sdma 29 4 0>, <&sdma 30 4 0>;
                dma-names = "rx", "tx";
                pinctrl-names = "default";
                pinctrl-0 = <&pinctrl_uart3>;
                status = "okay";
        };

        uart4{
                compatible = "fsl,imx6q-uart", "fsl,imx21-uart";
                reg = <0x021f0000 0x4000>;
                interrupts = <0 29 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_UART_IPG>,
                         <&clks IMX6QDL_CLK_UART_SERIAL>;
                clock-names = "ipg", "per";
                dmas = <&sdma 31 4 0>, <&sdma 32 4 0>;
                dma-names = "rx", "tx";
                pinctrl-names = "default";
                pinctrl-0 = <&pinctrl_uart4>;
                status = "okay";
        };

        uart5{
                compatible = "fsl,imx6q-uart", "fsl,imx21-uart";
                reg = <0x021f4000 0x4000>;
                interrupts = <0 30 IRQ_TYPE_LEVEL_HIGH>;
                clocks = <&clks IMX6QDL_CLK_UART_IPG>,
                         <&clks IMX6QDL_CLK_UART_SERIAL>;
                clock-names = "ipg", "per";
                dmas = <&sdma 33 4 0>, <&sdma 34 4 0>;
                dma-names = "rx", "tx";
                pinctrl-names = "default";
                pinctrl-0 = <&pinctrl_uart5_2>;
                status = "okay";
        };

        i2c1{
                compatible = "fsl,imx6q-i2c", "fsl,imx21-i2c";
                reg = <0x021a0000 0x4000>;
                interrupts = <0 36 IRQ_TYPE_LEVEL_HIGH>;
                clocks = &clks IMX6QDL_CLK_I2C1;
                status = "disabled";
        };

        i2c2{
                compatible = "fsl,imx6q-i2c", "fsl,imx21-i2c";
                reg = <0x021a4000 0x4000>;
                interrupts = <0 37 IRQ_TYPE_LEVEL_HIGH>;
                clocks = &clks IMX6QDL_CLK_I2C2;
```

```
                status = "disabled";
        };

        i2c3{
                compatible = "fsl,imx6q-i2c", "fsl,imx21-i2c";
                reg = <0x021a8000 0x4000>;
                interrupts = <0 38 IRQ_TYPE_LEVEL_HIGH>;
                clocks = &clks IMX6QDL_CLK_I2C3;
                status = "disabled";
        };

        i2c4{
                compatible = "fsl,imx6q-i2c", "fsl,imx21-i2c";
                reg = <0x021f8000 0x4000>;
                interrupts = <0 35 IRQ_TYPE_LEVEL_HIGH>;
                clocks = &clks IMX6DL_CLK_I2C4;
                status = "disabled";
        };

        usbh1{
                compatible = "fsl,imx6q-usb", "fsl,imx27-usb";
                reg = <0x02184200 0x200>;
                interrupts = <0 40 IRQ_TYPE_LEVEL_HIGH>;
                clocks = &clks IMX6QDL_CLK_USBOH3;
                status = "disabled";
        };

        usbh2{
                compatible = "fsl,imx6q-usb", "fsl,imx27-usb";
                reg = <0x02184400 0x200>;
                interrupts = <0 41 IRQ_TYPE_LEVEL_HIGH>;
                clocks = &clks IMX6QDL_CLK_USBOH3;
                status = "disabled";
        };

        usbh3{
                compatible = "fsl,imx6q-usb", "fsl,imx27-usb";
                reg = <0x02184600 0x200>;
                interrupts = <0 42 IRQ_TYPE_LEVEL_HIGH>;
                clocks = &clks IMX6QDL_CLK_USBOH3;
                status = "disabled";
        };

        usbotg{
                compatible = "fsl,imx6q-usb", "fsl,imx27-usb";
                reg = <0x02184000 0x200>;
                interrupts = <0 43 IRQ_TYPE_LEVEL_HIGH>;
                clocks = &clks IMX6QDL_CLK_USBOH3;
                status = "disabled";
        };

        caam{
                compatible = "fsl,sec-v4.0";
                address-cells = <1>;
                size-cells = <1>;
                reg = <0x2100000 0x40000>;
                ranges = <0 0x2100000 0x40000>;
                interrupt-parent = <&intc>;
                clocks = <&clks IMX6QDL_CAAM_MEM>, <&clks IMX6QDL_CAAM_ACLK>,
<&clks IMX6QDL_CAAM_IPG> ,<&clks IMX6QDL_CLK_EIM_SLOW>;
                clock-names = "caam_mem", "caam_aclk", "caam_ipg", "caam_emi_slow";

                jr0{
                        compatible = "fsl,sec-v4.0-job-ring";
                        reg = <0x1000 0x1000>;
                        interrupt-parent = <&intc>;
                        interrupts = <0 105 0x4>;
                };

                jr1{
                        compatible = "fsl,sec-v4.0-job-ring";
                        reg = <0x2000 0x1000>;
                        interrupt-parent = <&intc>;
                        interrupts = <0 106 0x4>;
                };
        };
```

```
                fec{
                        compatible = "fsl,imx6q-fec";
                        reg = <0x02188000 0x4000>;
                        interrupts-extended =
                                <&intc 0 118 IRQ_TYPE_LEVEL_HIGH>,
                                <&intc 0 119 IRQ_TYPE_LEVEL_HIGH>;
                        clocks = <&clks IMX6QDL_CLK_ENET>,
                                <&clks IMX6QDL_CLK_ENET>,
                                <&clks IMX6QDL_CLK_ENET_REF>;
                        clock-names = "ipg", "ahb", "ptp";
                        status = "disabled";
                };
        };
    };
};
```

Ek 3 Bilgilendirilmiş Onam Formu

LÜTFEN BU DÖKÜMANI DİKKATLİCE OKUMAK İÇİN ZAMAN AYIRINIZ

Sizi Sadık ARSLAN tarafından yürütülen "Aygıt Ağacı Yazılımlarının Model Güdümlü Geliştirilmesi" başlıklı **araştırmaya** davet ediyoruz. Bu araştırmaya katılıp katılmama kararını vermeden önce, araştırmanın neden ve nasıl yapılacağını bilmeniz gerekmektedir. Bu nedenle bu formun okunup anlaşılması büyük önem taşımaktadır. Eğer anlayamadığınız ve sizin için açık olmayan şeyler varsa, ya da daha fazla bilgi isterseniz bize sorunuz.

Bu çalışmaya katılmak tamamen **gönüllülük** esasına dayanmaktadır. Çalışmaya **katılmama** veya katıldıktan sonra herhangi bir anda çalışmadan **çıkma** hakkında sahipsiniz. **Çalışmayı yanıtlamanız, araştırmaya katılım için onam verdiğiniz** biçiminde yorumlanacaktır. Size verilen **formlardaki** soruları yanıtlarken kimsenin baskısı veya telkini altında olmayın. Bu formlardan elde edilecek kişisel bilgiler tamamen gizli tutulacak ve yalnızca araştırma amacı ile kullanılacaktır.

1. Araştırmayla İlgili Bilgiler:

a. Araştırmanın Amacı: Aygıt Ağacı (ing. Device Tree) (DT) yazılımlarının geliştirilmesi sırasında yazılım geliştiricilerin kullandığı alana-özgü modelleme dillerinin (ing. domain-specific modeling language) (DSML) ve bunlara bağlı yazılım araçlarının kullanılabilirliğininin değerlendirilmesi amaçlanmaktadır.

b. Araştırmanın İçeriği: DT yazılımların kodlanması öncesinde modellenmesine imkan veren DSML'lerin yazılım geliştiricilerin kullanımı açısından ne tür kolaylıklar getirdiği, klasik yazılım geliştirme süreçlerine göre varsa kullanım zorlukları ve üretkenlikleri (otomatik doküman ve kod üretimi) yazılım geliştiricilerin bu dilleri sistem geliştirmede kullanacakları durum çalışmaları üzerinden değerlendirilecektir. Araştırmaya gönüllü katılacak yazılım geliştiricilerin hiçbir kişisel / demografik bilgisi, sağlık bilgisi, görüntüsü, sesi vb. yürütülecek durum çalışmaları için gerekli değildir. Bu bilgiler çalışmada kullanılmayacak, anket verisi olarak tutulmayacak ve hiçbir yazılı metinde ve bilgisayar ortamında saklanmayacaktır. Bu geliştiricilerin sadece bilgisayar ortamında ilgili yazılım geliştirme araçlarını kullanarak ne kadar sürede yazılım geliştirdikleri ölçülecek ve üretilen yazılım kodlarının satır sayıları hesaplanacaktır.

c.

d.

2. Çalışmaya Katılım Onayı:

Yukarıda yer alan ve araştırmadan önce katılımcıya/gönüllüye verilmesi gereken bilgileri okudum ve katılmam istenen çalışmanın kapsamını ve amacını, gönüllü olarak üzerime düşen sorumlulukları tamamen anladım. **Çalışma hakkında yazılı ve sözlü açıklama aşağıda adı belirtilen araştırmacı tarafından yapıldı, soru sorma ve tartışma imkanı buldum ve tatmin edici yanıtlar aldım. Bana, çalışmanın muhtemel**

riskleri ve faydaları sözlü olarak da anlatıldı. Bu çalışmayı istediğim zaman ve herhangi bir neden belirtmek zorunda kalmadan bırakabileceğimi ve bıraktığım takdirde herhangi bir olumsuzluk ile karşılaşmayacağımı anladım.

Bu koşullarda söz konusu araştırmaya kendi isteğimle, hiçbir baskı ve zorlama olmaksızın katılmayı kabul ediyorum.

Katılımcının (Kendi el yazısı ile)
Adı-
Soyadı:.....
.....
İmzası:

(Varsa) Velayet veya Vesayet Altında Bulunanlar İçin;

Veli veya Vasisinin (kendi el yazısı ile)

Adı-
Soyadı:.....
.....
İmzası:

Not: *Bu form, iki nüsha halinde düzenlenir. Bu nüshalardan biri imza karşılığında gönüllü kişiye verilir, diğeri araştırmacı tarafından saklanır.*