



Cost Efficient Dependable Electronic Systems

Partitionerade applikationer i en distribuerad realtidsmiljö

Author	Fredrik Johansson, Olof Johnsson
Document Id	011
Date	December 2005
Availability Status	Public Final

CHALMERS



Design and Modelling of
Partitioned Applications in a
Distributed Real-time Environment

Partitionerade applikationer i en
distribuerad realtidsmiljö

FREDRIK JOHANSSON
OLOF JOHANSSON

Examensarbete

Civilingenjörsprogrammet för datateknik

CHALMERS TEKNISKA HÖGSKOLA
Institutionen för data- och informationsteknik
Avdelningen för datorteknik
Göteborg 2005

Innehållet i detta häfte är skyddat enligt Lagen om upphovsrätt, 1960:729, och får inte reproduceras eller spridas i någon form utan medgivande av författaren. Förbudet gäller hela verket såväl som delar av verket och inkluderar lagring i elektroniska och magnetiska media, visning på bildskärm samt bandupptagning.

© **Fredrik Johansson och Olof Johnsson**, Göteborg 2006.

Sammanfattning

I dagens fordonssystem har datorerna tagit en viktig roll med många olika arbetsuppgifter. När behovet för en ny uppgift framkommer så har man traditionellt sett valt att lägga till en datornod som sköter om den specifika uppgiften. Detta tillvägagångssätt blir dock alltmera opraktiskt ju fler uppgifter som behövs, och med nya x-by-wire tillämpningar så ser det ut att bli oöverskådligt många noder. För att komma tillrätta med detta problem så kan man ha flera tillämpningar körande på varje nod och på så sätt bättre använda varje datornods kapacitet. Framför allt ställer by-wire-tillämpningarna högre krav på tillförlitligheten än tidigare och feltolerans är ett måste i dessa säkerhetskritiska system. Detta examensarbete har som syfte att skapa tre olika tillämpningar som modelleras och simuleras för att sedan användas i framtida experiment inom partitionerade system och redundanshantering. Modellen består av sex noder med de sensorer och aktuatorer som krävs från tillämpningarna. Kravet är att vilken som helst nod, sensor eller aktuator när som helst skall kunna falla och tillämpningarna skall då fortfarande kunna leverera en felfri funktion, detta åstadkoms genom att låta tillämpningarna migrera till andra fungerande noder. Vi har visat att detta är en fungerande lösning, men att mycket återstår för att det skall fungera i praktiken.

Sökord: Feltoleranta system, applikationsmigrering, partitionerade applikationer, membership agreement, by-wire-modeller, redundanshantering, realtidssystem.

Abstract

With lots of different tasks, the computers in modern cars are key components. When the need for additional tasks arises, the traditional method of choice has been to simply add a new computer node. With this approach the number of nodes needed gets immense, but by letting different tasks share the same node this problem can be solved. This demands higher reliability and fault tolerance from the tasks that are safety critical. The purpose of this master thesis is to create different tasks that are modelled and simulated for the use in future experiments in the area of partitioned systems, redundancy management and membership agreement. Our model consists of six nodes with the necessary actuators and nodes. It is a requirement that the task delivers correct functionality despite that any one of the nodes, sensors or actuators may break down. This is achieved with task migration, where the tasks start up at other working nodes. We have shown this to be a possible solution, but there is a lot of more work to be done before it is working in practice.

Keywords: Fault tolerant systems, task migration, partitioned applications, membership agreement, by-wire models, redundancy management, real-time system.

Förord

Detta är ett examensarbete i Datalogi vid Institutionen för Data-teknik på Chalmers tekniska högskola med examinatorn Arne Dahlberg.

Arbetet utförs på Mecel för handledarna dr Håkan Sivencrona (Mecel) och Carl Bergenhem (SP) som en del av GAST-projektet.

Innehållsförteckning

1.	Inledning	7
1.1.	Bakgrund	7
1.2.	Syfte	7
1.3.	Omfattning	8
1.4.	Avgränsningar	8
1.4.1.	Allmänt	8
1.4.2.	BBW-applikationen	9
1.4.3.	SBW-applikationen	9
1.4.4.	SBB-applikationen	9
1.5.	Angränsande projekt	9
2.	Systemdokumentation	10
2.1.	Övergripande systemspecifikation	10
2.1.1.	Miljön	10
2.1.2.	Noderna	10
2.2.	Krav	12
2.2.1.	Sensorerna	12
2.2.2.	Aktuatorerna	12
2.2.3.	Tider	13
2.3.	Applikationerna	14
2.3.1.	Broms	14
2.3.2.	Styrning	20
2.3.3.	Styrning med broms	22
2.3.4.	Middleware	24
2.4.	Simulink-modell	28
3.	Simuleringsresultat	36
4.	Analys	42
4.1.	Tillförlitlighet	42
4.2.	Problem	44
5.	Slutsats	46
5.1.	Framtida arbete	46
6.	Förkortningar	47
6.1.	Komponenter	47
6.2.	Applikationer	47
6.3.	Testfall	47
7.	Referenser	48

1. Inledning

1.1. Bakgrund

I dagens fordon sköts styrning med hjälp av mekanik, exempelvis bromsning där överföringen sker med hydraulik. Att låta överföringen ske på detta sätt innebär många problem, bland annat kommer användningen av rörliga, mekaniska system oundvikligen att leda till slitage. Det skapar problem för konstruktörerna som måste ta hänsyn till den mekaniska överföringen när de konstruerar fordonet. Även tillverkningen av fordonet försåras av att använda mekanisk överföring eftersom det innebär tunga lyft och arbete i trånga utrymmen för montörerna.

I moderna fordon finns hjälpsystem såsom servobroms och servostyrning för att underlätta för föraren. Att på detta sätt ha dubbla styrsystem ökar risken för fel i fordonet.

För att råda bot på dessa problem introduceras därför elektronisk överföring i fordon i allt större utsträckning. I dagsläget är det endast icke säkerhetskritiska funktioner som helt och hållet sköts med elektronik. Men mycket finns att vinna på att helt övergå till elektronisk kommunikation i fordon.

För de elektroniska system som existerar idag finns dock många förbättringar som kan göras. De består nämligen av en hel mängd olika noder, i en normal bil ca 70 noder [1], där varje nod har sin egen specifika uppgift. Om sen dessa noder behöver kommunicera med andra noder kommer antalet sammankopplingar snart att växa explosionsartat och därmed ge upphov till tjocka kabelbuntar som på något sätt skall dras mellan noderna i fordonet. För att komma till rätta med dessa existerande problem med dagens elektroniska överföring behövs ett nytt system för kommunikation. Detta nya system innebär att man ökar noderernas beräkningskapacitet och låter varje enskild nod sköta fler än en uppgift. På detta sätt kan antalet noder drastiskt minskas. Förhoppningsvis kommer inte fler än ca 20 noder vara nödvändiga [6]. Kommunikationen kommer att ske via databussar vilket leder till att kabelbuntarna, som förut behövdes, minskas till ett minimum. När systemet är byggt på detta sätt kan man på allvar tala om "drive-by-wire" [11], även kallat "X-by-wire".

All nödvändig teknik för att utforma kommunikationen i fordon på detta sätt existerar idag, men det största hindret återstår ännu. För att kunna använda elektronisk kommunikation för säkerhetskritiska system krävs felsäkerhet och, kanske ännu viktigare, att användarna är övertygade om att fordonet är pålitligt.

1.2. Syfte

Detta examensarbete är en del av GAST-projektet [8] som pågått under ett par års tid. GAST-projektet har som mål att utveckla hård- och mjukvara för distribuerade, inbyggda realtidssystem. Produkterna (GAST-noderna) kommer bland annat att lämpa sig mycket väl för utveckling av fordonsnoder. Dessa

noder stödjer ett flertal kommunikationsprotokoll, bland annat höghastighetskommunikation med Flexray [7]. Applikationer till GAST-noderna skall byggas för framtida experiment inom partitionerade system och skall möjliggöra redundanshantering och task migration (applikationerna flyttas över till en annan nod). För att få en snabb utveckling av ny mjukvara till dessa noder skall utvecklingsarbetet till så stor del som möjligt ske med modeller. Från modellerna skall körbar kod kunna genereras. Som en start på arbetet med denna utvecklingsstrategi skall några testapplikationer utvecklas i form av modeller. Dessa modeller skall simuleras och slutligen skall kod genereras.

1.3. Omfattning

Tre applikationer skall modelleras och simuleras – BBW (Brake By Wire), SBW (Steer By Wire) och SBB (Steer By Brake). För att dessa tjänster skall fungera skall även en extra applikation per nod modelleras, nodens s.k. MV (middleware). Middleware är alltså den applikation som möjliggör task migration.

För arbetet med att skapa applikationsmodellerna behöver en miljö innehållande 6 noder med tillhörande sensorer och aktuatorer definieras. Applikationerna skall leverera en felfri funktion (dock i vissa fall en degraderad version) även då vilken som helst nod, aktuator eller sensor fallerar. Detta kan endast åstadkommas genom att tillåta att applikationerna migrerar mellan noderna.

För att skapa modeller av systemet har vi valt att använda Simulink [10]. Detta val gjordes i första hand för att vi har förkunskaper om det systemet men även för att SCADE [9], som var det andra systemet vi funderade på att använda, var svårare att få tag i samt mycket dyrare.

1.4. Avgränsningar

1.4.1. Allmänt

- Applikationerna behöver inte ta någon hänsyn till realtidsschemaläggning.
- Att styrvärden ligger inom givet intervall när de når hjulnodsapplikationerna är ett förvillkor.
- Applikationerna behöver inte ta hand om membership agreement. Middleware kommer att få tillgång till aktuell status på de noder, sensorer och aktuatorer som existerar i vår miljö.
- Endast ett fel behöver tas om hand. Om fler än ett fel inträffar samtidigt kan funktion ej garanteras.
- Varje enskild nod är endast kopplad till vissa av sensorerna och aktuatorerna i miljön. Att koppla ihop samtliga noder med samtliga sensorer och aktuatorer skulle visserligen ge ett system med mycket större frihet, men för vårt arbete är detta inte nödvändigt och vi har därför valt att minska komplexiteten på detta sätt. Hur miljön ser ut finns ytterligare specificerat i kapitel 6.1.

- Vi antar att inga fel inträffar i datakommunikationsbussarna.
- Vi har ansett att om det beräknade värdena skiljer sig från varandra med mindre än 10^{-10} så ligger det inom felmarginalen och bedöms därför som likadana.
- Valet att bara dubblera sensorerna för broms och styrning (S_B och S_W) baseras på att dessa är säkerhetskritiska.
- Broms- och styrvärden skickas med jämna mellanrum från NHMI eller NM till hjulnoderna, även då värdet är 0. Detta innebär mycket onödigt trafik på bussen, men vi räknar med att vi ändå har en egen tidslucka som kan utnyttjas.
- På de ställen i modellen där samma beräkning kan utföras summeras resultatet eftersom bara en åt gången kan vara aktiv.

1.4.2. BBW-applikationen

- Hjulnoderna kontrollerar endast sitt eget beräknade värde. Därför upptäcks inte andra noders fel och noden kan således ej stoppa en felaktig nod som tror sig räkna rätt.

1.4.3. SBW-applikationen

Ingen avgränsning för endast denna applikation.

1.4.4. SBB-applikationen

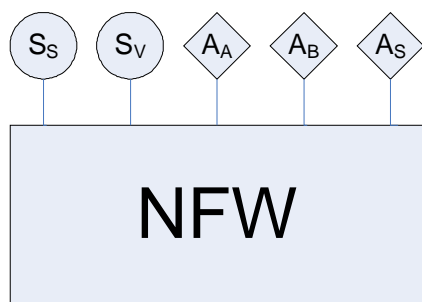
- Styrning kommer endast att ske genom bromsning av framhjulen eftersom det inte skulle ge något mervärde för vårt projekt att kunna styra genom att bromsa även med bakhjulen.
- Vi kan inte kontrollera om framhjulet beräknar rätt värde eftersom det bara är ett av dem som bromsar extra beroende på vilket håll som det skall styras åt.
- Då utstyrningen av hjulen slutar fungera utgår vi från att hjulen återgår till att vara riktade rakt fram i fordonets färdriktning.

1.5. *Angränsande projekt*

Detta examensarbete är ett av många som utförs under GAST-projektet. Det tidigare arbetet i projektet har lett fram till att det nu finns fungerande noder. Arbetet som vi varit en del av och som pågått parallellt med vårt examensarbete syftar till att skapa ett fungerande testsystem, där ett antal noder kopplats samman och fungerar tillsammans i realtid, med funktioner såsom task migration och membership agreement.

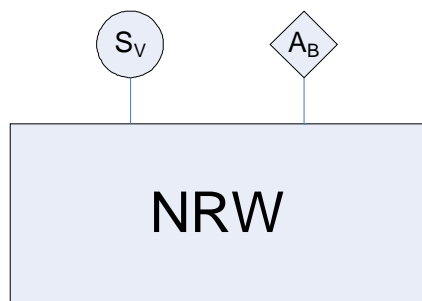
- S_S är sensorn som känner av vilken vinkel hjulet har.
- S_V är sensorn som känner av vilken hastighet hjulet har.
- A_A är aktuatoren som sköter drivningen på hjulet.
- A_B är aktuatoren som sköter bromsen på hjulet.
- A_S är aktuatoren som sköter hjulets vridning.

Att det finns möjlighet att ha olika vinkel och hastighet på de båda framhjulen beror på att det ger extra möjligheter att styra bilen på precis de sätt som för tillfället är de optimala.



Figur 2-2 – Framhjulsnod

Figur 2-3 visar en översikt av de bakre hjulnoderna. Dessa har en sensor och en aktuator kopplade till sig, S_V som är hastighets-sensorn för hjulet samt A_B som bromsar hjulet. Bakhjulen skall inte ha någon drivning eller möjlighet att vridas och därför behövs inga fler sensorer eller aktuatorer. S_V finns på bakhjulen för att få en referens från ett icke drivande hjul för att kunna märka om och när de drivande framhjulen spinner.

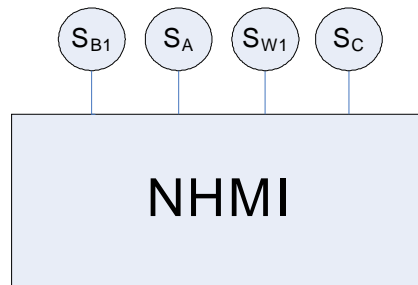


Figur 2-3 - Bakhjulsnod

Figur 2-4 visar en översiktsbild av den nod som kontrollerar förarmiljön.

- S_A är sensorn som känner av med vilken kraft gaspedalen trycks ner.
- S_{B1} är sensorn som känner av med vilken kraft bromspedalen trycks ner.
- S_{W1} är sensorn som känner av rattutslaget.
- S_C är sensorn som känner av ifall farthållaren är inkopplad eller inte.

Anledningen till att S_{B1} och S_{W1} är numrerade är att även NM är kopplad till ett par sensorer med samma uppgift, S_{B2} samt S_{W2} .

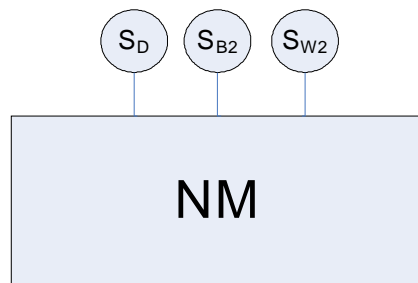


Figur 2-4 – Förarmiljönoden

Figur 2-5 visar en översiktsbild av huvudnoden. Tre sensorer är kopplade till noden.

- S_D är en accelerationssensor för att kunna mäta accelerationen i sidled.
- S_{B2} är sensorn som känner av med vilken kraft bromspedalen trycks ner.
- S_{W2} är sensorn som känner av rattutslaget.

Anledningen till att vi har en sån sensor är för att kunna häva sladdar. De andra sensorerna är backupsensorer som är samma som i NHMI.



Figur 2-5 – Huvudnoden

2.2. Krav

För att applikationerna skall kunna leverera given funktion finns vissa krav som måste vara uppfyllda.

2.2.1. Sensorerna

- Sensorvärdet från S_{B1} och S_{B2} skall vara ett värde mellan 0 och 1, där 0 är det minsta (dvs. ingen bromskraft) och 1 är det största (dvs. full bromskraft).
- Sensorvärdet från S_{W1} och S_{W2} skall vara ett värde mellan -1 och 1, där -1 är fullt rattutslag åt vänster, 1 är fullt rattutslag åt höger och 0 är inget rattutslag alls.

2.2.2. Aktuatorerna

- De värden som skickas till A_{BFL} , A_{BFR} , A_{BRL} och A_{BRR} kommer vara värden mellan 0 och 1, där 0 är det minsta (dvs. ingen bromskraft) och 1 är det största (dvs. full bromskraft). Dessa värden måste aktuatoren kunna tolka korrekt och generera rätt bromskraft.

- De värden som skickas till A_{SFL} och A_{SFR} kommer att vara värden mellan -1 och 1, där -1 är full vridning åt vänster, 1 är full vridning åt höger och 0 är ingen vridning alls. Dessa värden måste aktuatoren kunna tolka korrekt och generera rätt vridning på hjulen.

2.2.3. Tider

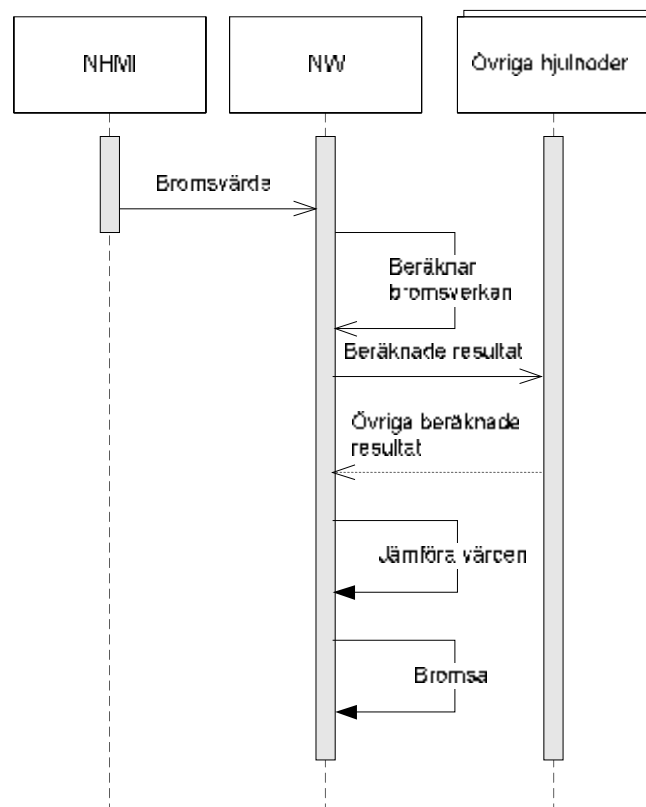
Enligt ett pågående examensarbete på Chalmers [4] måste korrigering av aktuatorer ske med minst 10 ms mellanrum. Vissa av våra applikationer behöver kommunicera två gånger. Dessutom kommer själva beräkningarna i applikationerna att ta en viss tid, vi har i vårt arbete antagit att beräkningarna tar 3 ms. Följande krav gäller därför:

- Meddelanden som skickas mellan applikationer måste nå mottagaren inom maximalt 2 ms, detta krav gäller även då applikationerna körs på olika noder.
- Uppdaterade sensorvärden måste finnas tillgängliga med maximalt 10 ms mellanrum.

2.3. Applikationerna

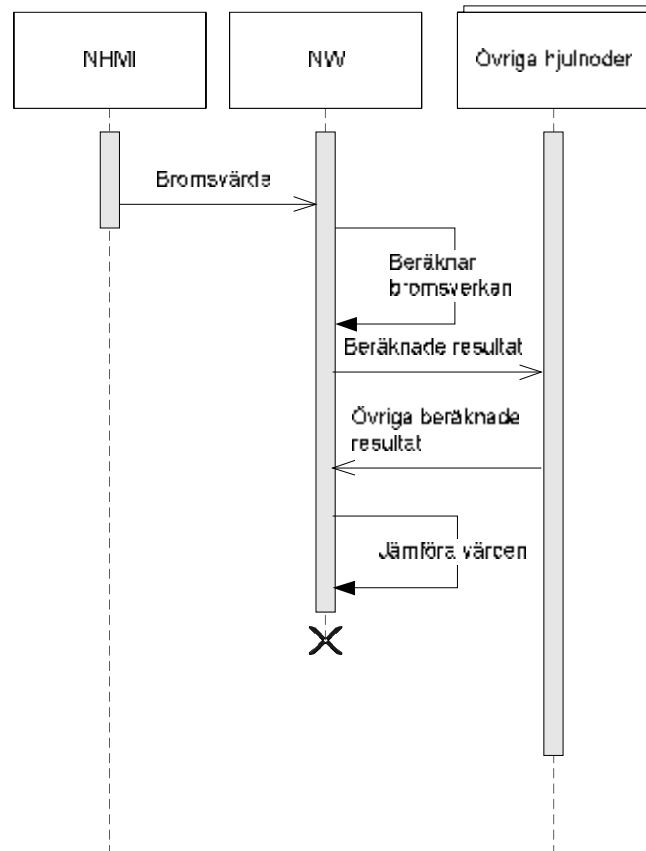
2.3.1. Broms

Diagrammet i figur 2-6 förtydligar kommunikationen och de huvudsakliga uppgifterna som utförs i någon utav hjulnoderna. I detta fall fungerar allt som det skall. Först skickas den önskade bromsverkan från NHMI till hjulnoden. Där beräknas hur mycket det egna hjulet skall bromsas men även bromsverkan för de övriga hjulen. Detta skickas vidare så att alla hjulnoder kan ta del av resultatet och jämföra med sitt egna beräknade värde. I detta fall så överensstämmer värdena och bromsning kan ske på normalt vis genom att aktivera aktuatoren på respektive hjul.



Figur 2-6 - Aktivitetsdiagram för bromsning vid korrekt beräknade resultat

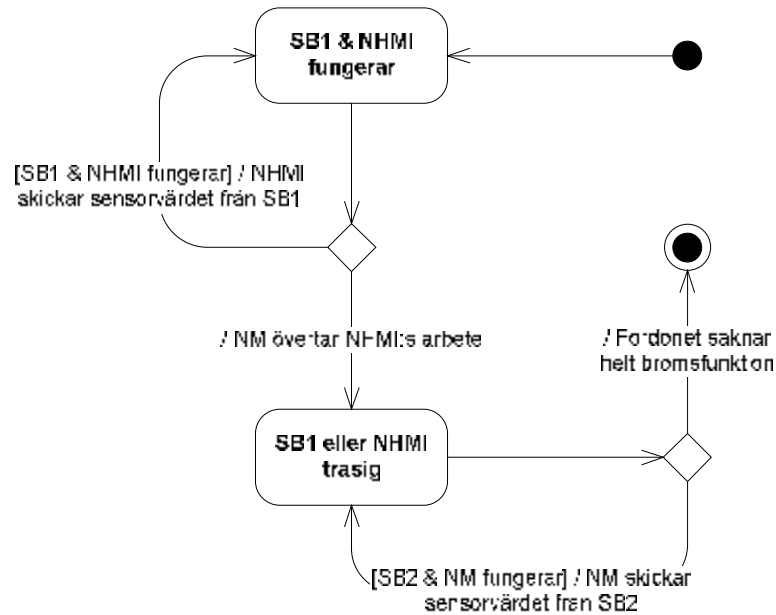
I detta fall, figur 2-7, så överensstämmer inte det beräknade värdet hos hjulnoden med de övriga hjulnodernas värden. Detta resulterar i att den aktuella hjulnoden slutar att svara på förfrågningar och man får nöja sig med att kunna bromsa på två hjul. Detta beror på att vi också stänger av motstående nod på samma axel p.g.a. att vi inte vill ha några roterande moment vid bromsning.



Figur 2-7 - Aktivitetsdiagram för bromsning vid inkorrekt beräknade resultat

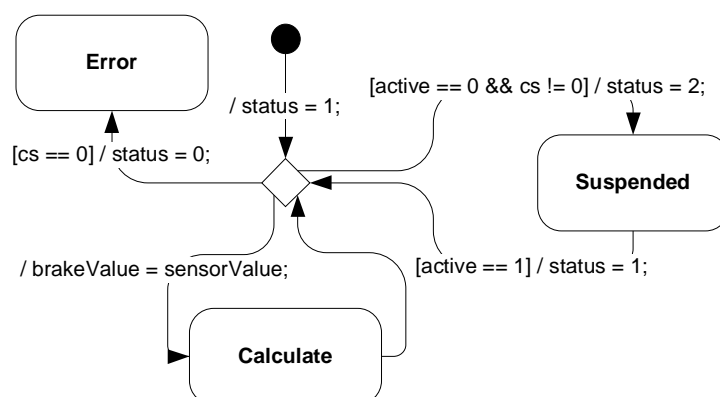
Diagrammet i figur 2-8 visar flödet för bromsens sensorfunktion som innefattar sensorerna S_{B1} och S_{B2} samt noderna NHMI och NM. NM används som backupnod för att öka felsäkerheten.

Systemet består av två tillstånd – ” S_{B1} & NHMI fungerar” och ” S_{B1} eller NHMI trasig”. Så länge S_{B1} och NHMI fungerar används NHMI för att läsa av S_{B1} :s sensorvärde och skicka detta till hjulnoderna. Om S_{B1} eller NHMI av någon anledning skulle sluta fungera så kommer uppgiften att övertas av NM. NM använder sig av sensorn S_{B2} och skickar sedan ut sensorvärdet till hjulnoderna.



Figur 2-8 – Funktionsöversikt av NHMI och NM

Figur 2-9 visar hur bromsapplikationen som finns på noderna NHMI och NM fungerar. Applikationen består av tre tillstånd – ”Calculate”, ”Suspended” samt ”Error”. Vid ett fullt fungerande system så kommer sensorvärdet att läsas av NHMI, därför kommer då NHMI att befinna sig i ”Calculate”-tillståndet. Om någonting inträffar så att applikationen inte längre kan köras på NHMI, t ex om S_{B1} slutar fungera, kommer applikationen som körs på NHMI att hamna i ”Error”-tillståndet. Samtidigt kommer applikationen som körs på NM att gå ur ”Suspended”-tillståndet och istället hamna i ”Calculate”-tillståndet. Den stannar i ”Calculate”-tillståndet tills något inträffar som innebär att applikationen inte längre kan köras på NM, t ex om S_{B2} slutar fungera, och hamnar då i tillståndet ”Error”. I detta läge har fordonet inte längre någon bromsfunktion.



Figur 2-9 – Applikationen på NHMI och NM

Bromsfunktionen i hjulnoderna består i stora drag av fyra olika tillstånd – ”en eller två aktiva noder”, ”tre aktiva noder”, ”fyra aktiva noder” samt ”oanvänd”, se figur 2-10. Tillstånden ”fyra aktiva noder” och ”en eller två aktiva noder” har även var sitt lokalt tillstånd – ”Väntar på resultat”.

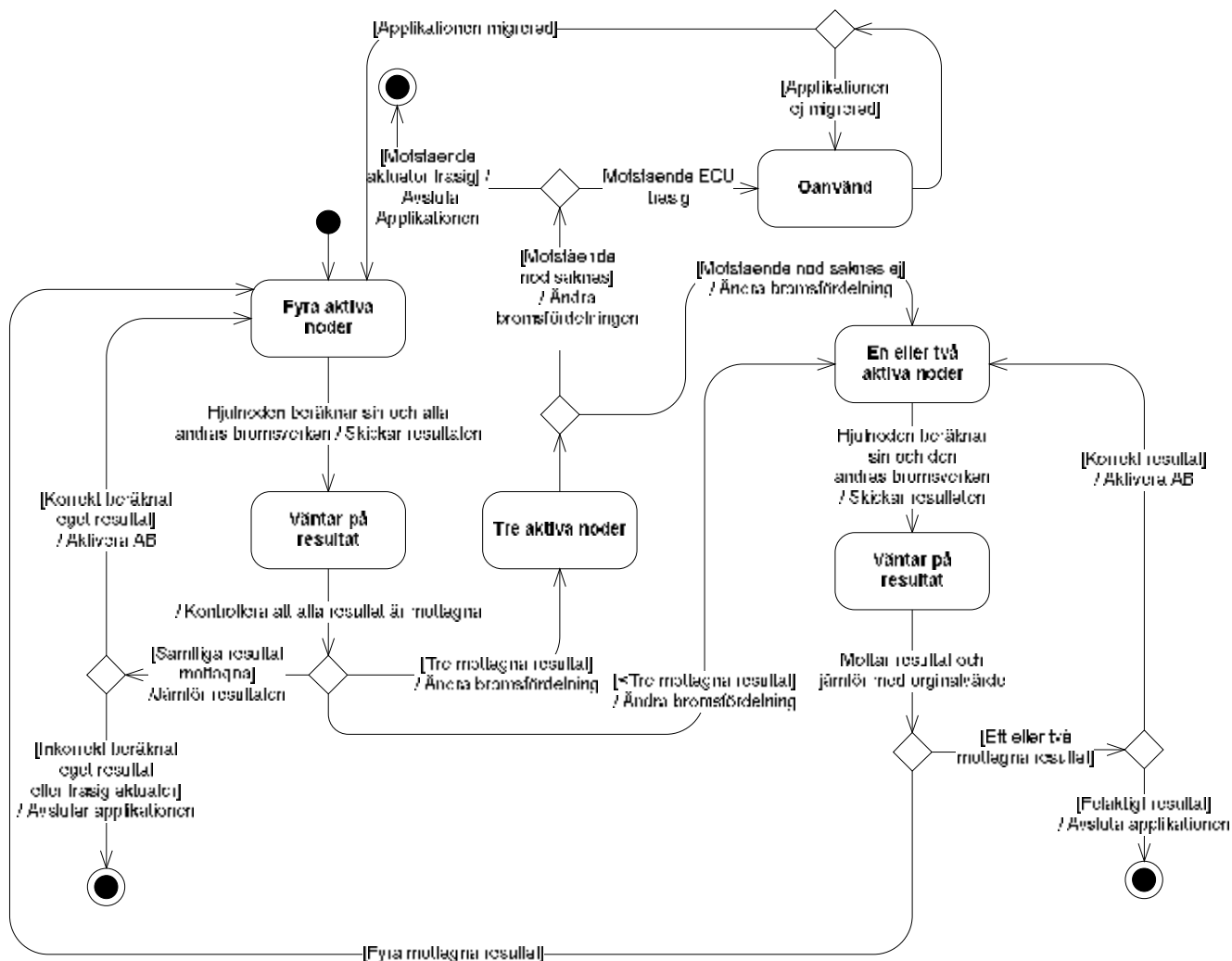
I startskedet finns fyra aktiva noder och tillståndet ”fyra aktiva noder” är därmed aktivt. Varje nod mottar ett bromsvärde från NHMI och beräknar då sin egen samt alla övriga noders bromsverkan. Resultaten skickas på bussen till de övriga hjulnoderna och applikationen hamnar sedan i tillståndet ”väntar på resultat”. När applikationen lämnar tillståndet ”väntar på resultat” så kontrolleras om värden från samtliga övriga noder har mottagits. Om så inte är fallet kommer tillståndet ändras till det nu gällande, d.v.s. har tre resultat blivit mottagna ändras tillståndet till ”tre aktiva sensorer”, ett eller två mottaget resultat ger tillståndet ”en eller två aktiva noder”. I annat fall, d.v.s. när samtliga värden tagits emot, kommer det egna beräknade värdet jämföras med de värden som de övriga noderna beräknat för den aktuella noden. Om det egna beräknade värdet, genom röstning, visar sig vara felaktigt så avslutas applikationen. I annat fall aktiveras bromsen och det aktiva tillståndet är fortfarande ”fyra aktiva noder”.

Tillståndet ”tre aktiva noder” har till uppgift att se till att fordonet inte bromsar ojämnt när endast ett hjul bromsar. När ett av hjulen inte längre kan bromsas, p.g.a. trasig aktuator, kommer

därför även den andra noden på samma hjulaxel att avslutas. Om däremot hjulet inte kan bromsas p.g.a. trasig nod, kommer den andra noden på samma axel bli oanvänd fram till dess att migrering av applikationen på den trasiga noden har skett. Exempelvis om vänster framhjuls bromsaktuator inte längre är funktionsduglig så kommer även bromsapplikationen på höger framhjulsnod att avslutas. Det hjulnodspar där båda noderna fortfarande är aktiva kommer att gå direkt vidare till tillståndet ”en eller två aktiva noder”.

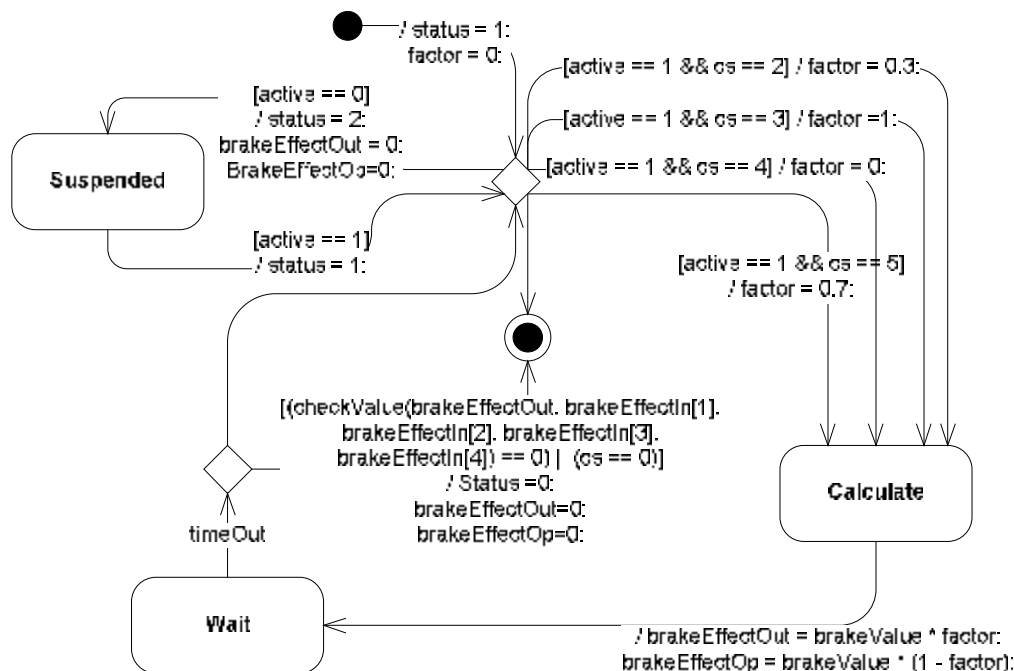
Tillståndet ”en eller två aktiva noder” fungerar enligt samma princip som tillståndet ”fyra aktiva noder”. Den aktuella noden beräknar alltså värdena för sin och den andra aktiva noden och skickar sedan resultatet. Noden väntar sedan på resultatet från den andra noden i tillståndet ”väntar på resultat”. I fallet med endast två aktiva noder kan inte röstning användas. Men eftersom man i detta tillstånd bromsar med full verkan på båda de kvarvarande hjulen så kan man jämföra det beräknade värdet med det som skickades från NHMI, är beräkningen korrekt bör de vara lika. Om så är fallet aktiveras bromsen och tillståndet återgår till ”en eller två aktiva noder” annars avslutas den felaktiga applikationen.

Om inget resultat tas emot från den andra noden kommer endast ett av hjulen att bromsas i väntan på att den andra noden skall migreras.



Figur 2-10 – Funktionsöversikt av hjulnoderna

Figur 2-11 visar hur applikationen på hjulnoderna fungerar. Denna applikation består av tre tillstånd – ”Calculate”, ”Wait” och ”Suspended”. ”Calculate”-tillståndet sköter själva beräkningen av bromsvärdet genom att multiplicera brakeValue (som är invärdet från sensornoden) med factor (som är en faktor som sätts av MW för att få olika bromsverkan i olika moder). Denna beräkning görs även för de hjulnoder som sitter på den andra axeln (brakeEffectOp). I ”Wait”-tillståndet väntar programmet för att övriga noder skall hinna beräkna och skicka sina bromsvärden. Dessa värden jämförs sedan med det värde som den aktuella noden beräknat för sig själv. Om det beräknade värdet genom röstning bedöms som felaktigt kommer applikationen att avslutas.



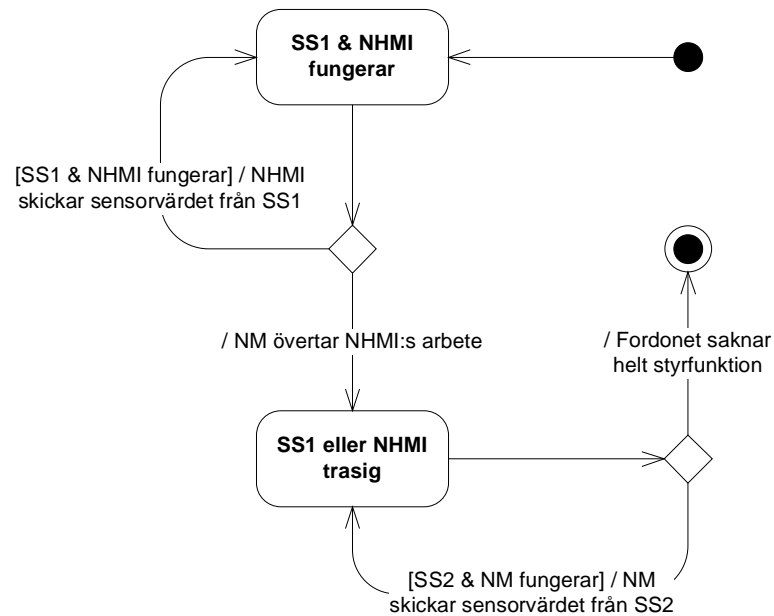
Figur 2-11 – Applikationen på hjulnoderna

2.3.2. Styrning

Figur 2-12 visar en översikt över funktionaliteten för att hämta sensorvärdet från rattsensorn (S_{W1} eller S_{W2}) och skicka detta värde till de hjulnoder som kör styrapplikationen (NFL och NFR).

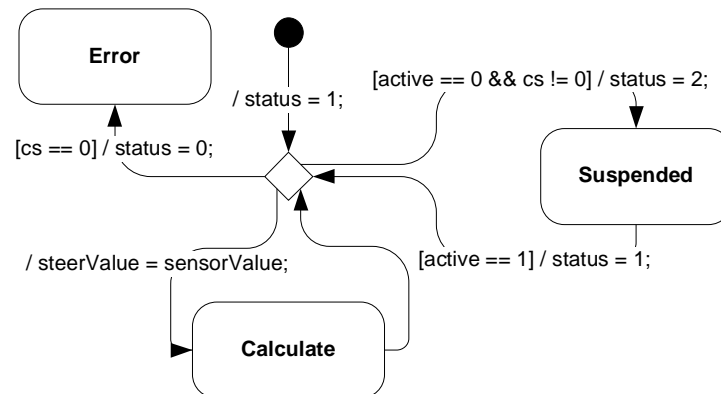
Två tillstånd existerar i denna modell – ” S_{W1} & NHMI fungerar” och ” S_{W1} eller NHMI trasig”. Så länge både S_{W1} och NHMI fungerar så kommer dessa att användas för att läsa och skicka värdet som anger rattutslag till de svängande hjulnoderna. Men då antingen S_{W1} eller NHMI inte längre fungerar så kommer funktionen att läsa och skicka rattutslagsvärdet att migreras till NM som använder sig av sensorn S_{W2} . S_{W1} och S_{W2} har samma funktionalitet och är endast dubblerade för att öka feltoleransen i systemet.

Om S_{W2} eller NM slutar fungera då tillståndet ” S_{W1} eller NHMI trasig” är aktivt så kommer fordonet inte längre ha någon möjlighet att styra.



Figur 2-12 – Funktionsöversikt av NHMI och NM

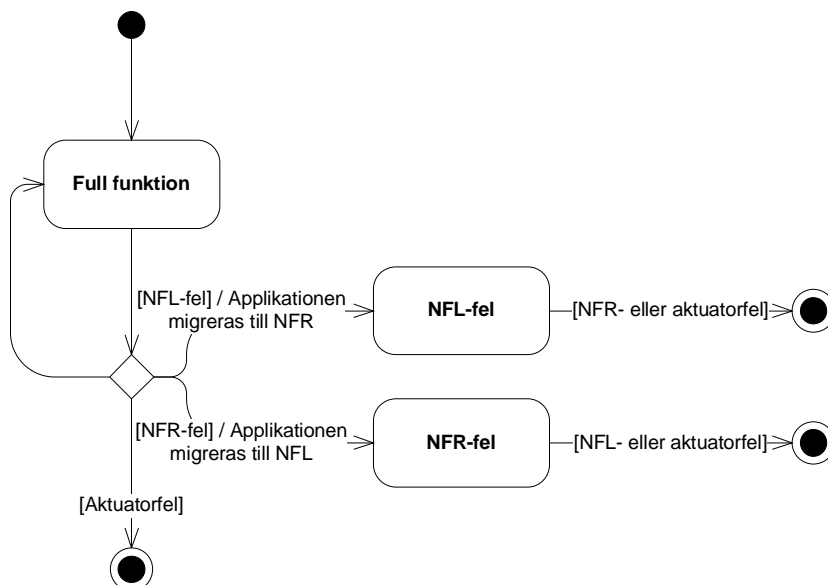
Applikationen för styrning, som körs på både NHMI och NM, visas i figur 2-13. Applikationen består av tre tillstånd – ”Calculate”, ”Suspended” och ”Error”. NM kommer att starta med active-värdet 0 och kommer därför att vara i ”Suspended”-tillståndet. NHMI kommer däremot att vara aktiv och därmed befinna sig i ”Calculate”-tillståndet fram tills cs-signalen ändras till 0, vilket innebär att NHMI eller SS1 inte fungerar och att applikationen då skall avslutas. Samtidigt som applikationen avslutas på NHMI så startas den på NM. Styrapplikationen migrerar från NHMI till NM.



Figur 2-13 – Applikationen på NHMI och NM

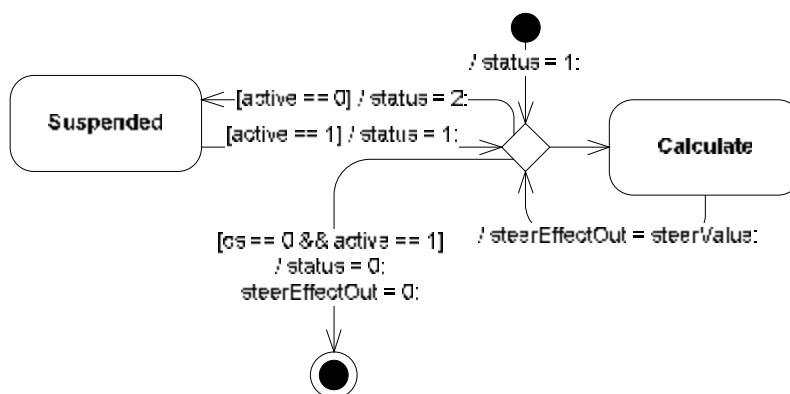
I figur 2-14 visas en översikt över styrfunktionaliteten på framhjulsnoderna. Från början fungerar alla delar som krävs för funktionen och systemet befinner sig därför i tillståndet ”Full funktion”. Om någon av aktuatorerna (A_{SFL} eller A_{SFR}) slutar fungera så kommer styrfunktionaliteten att avslutas, istället kommer applikationen för ”styrning med broms” att överta styrfunktionen. Om istället någon av noderna (NFL eller NFR) slutar fungera så kommer motsvarande tillstånd att bli aktivt (”NFL-fel” eller ”NFR-fel”) vilket innebär att den nod som fortfarande fungerar kommer att överta funktionaliteten från den

nod som slutat fungera. Om även den andra noden slutar fungera kommer bilen inte längre ha någon styrfunktion. Slutar däremot någon av aktuatorerna att fungera så kommer även här ”styrning med broms”-applikationen att överta styrfunktionen.



Figur 2-14 – Funktionsöversikt av framhjulsnoderna

Applikationen för styrning som körs på framhjulsnoderna är uppbyggd enligt figur 2-15. Denna applikation består alltså av två tillstånd – ”Calculate” och ”Suspended”. ”Suspended”-tillståndet kommer endast att användas för den applikation som är avsedd för att ersätta den andra noden (ex. den applikation som finns på NFL och ersätter applikationen som finns på NFR då denna nod inte längre fungerar). ”Calculate”-tillståndet är det normala tillståndet för en aktiv applikation. Där utförs den beräkning som ändrar styrvärdet som kommer från NHMI eller NM till den styreffekt som skickas till den aktuella nodens styraktuator.

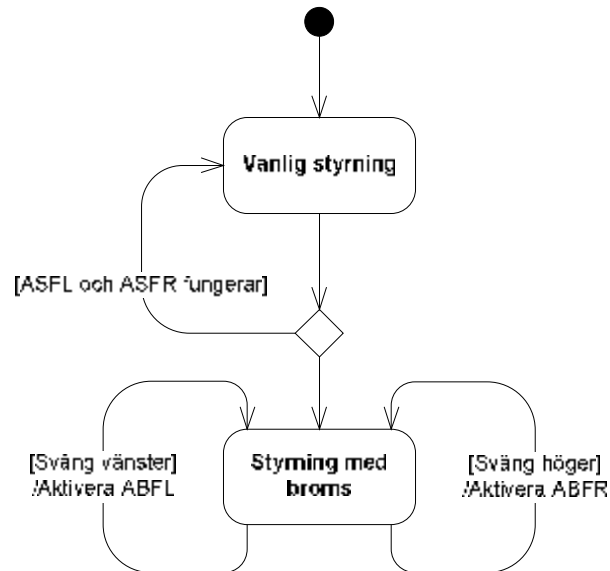


Figur 2-15 – Applikationen på framhjulsnoderna

2.3.3. Styrning med broms

NHMI och NM behöver ingen extra applikation eftersom styrapplikationen kan användas.

Figur 2-16 visar hur ”styrning med broms”-applikationen fungerar. Så länge A_{SFL} och A_{SFR} fungerar som de skall så sköts styrningen med hjälp av dem, men om någon av dessa slutar fungera så sköts styrningen istället med hjälp av A_{BFL} och A_{BFR} . När fordonet skall svänga till vänster aktiveras A_{BFL} och när fordonet skall svänga till höger aktiveras A_{BFR} .



Figur 2-16 – Funktionsöversikt av framhjulsnoderna

Figur 2-17 visar applikationen som körs på hjulnoderna för styrning med hjälp av bromsen. Denna applikation ersätter bromsapplikationen. Därför finns all funktion från bromsapplikationen kvar men med vissa tillägg. De gråa delarna är de som blivit tillagda.

Ett par nya insignaler (”place” och ”turn”) samt en ny intern variabel (”turnValue”) har tillkommit.

- Insignalen ”place” avgör vilken nod applikationen är placerad på (1 för NFL och 2 för NFR).
- Insignalen ”turn” är styrvärdet som visar hur mycket fordonet skall svänga (ett värde mellan -1 och 1).
- Variabeln ”turnValue” är styrvärdet omvandlat till ett bromsvärde.

Med hjälp av en ny valbox i applikationen kommer endast vänsterhjulet bromsas då föraren vill svänga åt vänster och vice versa.

I tabell 2-1 visas händelsematrisen för bromsfunktionen. Varje kolumn representerar en nod och varje rad de delar i våran miljö som kan sluta fungera. MW i NFL arbetar alltså enligt NFL-kolumnen, MW i NFR enligt NFR-kolumnen, osv.

Tabell 2-1 - Händelsematris för BBW

Aktuell \ Trasig	NFL	NFR	NRL	NRR	NHMI	NM
NFL	-	Starta upp NFLs applikation på egen nod.	-	-	-	-
ABFL	Ändra bromsverkan till 0%, behåll beräkningarna	Ändra bromsverkan till 0%, behåll beräkningarna	Ändra bromsverkan till 100%	Ändra bromsverkan till 100%	-	-
NFR	Starta upp NFRs applikation på egen nod.	-	-	-	-	-
ABFR	Ändra bromsverkan till 0%, behåll beräkningarna	Ändra bromsverkan till 0%, behåll beräkningarna	Ändra bromsverkan till 100%	Ändra bromsverkan till 100%	-	-
NRL	-	-	-	Starta upp NRLs applikation på egen nod.	-	-
ABRL	Ändra bromsverkan till 100%	Ändra bromsverkan till 100%	Ändra bromsverkan till 0%, behåll beräkningarna	Ändra bromsverkan till 0%, behåll beräkningarna	-	-
NRR	-	-	Starta upp NRRs applikation på egen nod.	-	-	-
ABRR	Ändra bromsverkan till 100%	Ändra bromsverkan till 100%	Ändra bromsverkan till 0%, behåll beräkningarna	Ändra bromsverkan till 0%, behåll beräkningarna	-	-
NHMI	-	-	-	-	-	Starta applikation för bromsning med SB2
SB1	-	-	-	-	Avsluta applikation	Starta applikation för bromsning med SB2
SB2	-	-	-	-	-	-
NM	-	-	-	-	-	-

Tabell 2-2 visar händelsematrisen för styrapplikationen. Bak-hjulen är inte representerade eftersom de ej kan svänga.

Tabell 2-2 - Händelsematris för SBW

Aktuell Trasig \	NFL	NFR	NHMI	NM
NFL	Avsluta applikation	Starta upp NFLs applikation på egen nod.	-	-
ASFL	Avsluta applikation	Avsluta applikation	-	-
NFR	Starta upp NFRs applikation på egen nod.	Avsluta applikation	-	-
ASFR	Avsluta applikation	Avsluta applikation	-	-
NHMI	-	-	Avsluta applikation	Starta applikation för styrning med SW2
SS1	-	-	Avsluta applikation	Starta applikation för styrning med SW2
SS2	-	-	-	Avsluta applikation
NM	-	-	-	Avsluta applikation

I tabell 2-3 så ser vi händelsematrisen för styrning med broms-applikationen. Eftersom ordinarie styrapplikation används av NHMI och NM så behövs ingen speciell händelsematris. Bak-hjulen är inte påverkade av styrningen därför finns inte heller dessa med i matrisen.

De ut signaler som MW genererar finns i tabell 2-4.

Tabell 2-3 - Händelsematris för SBB

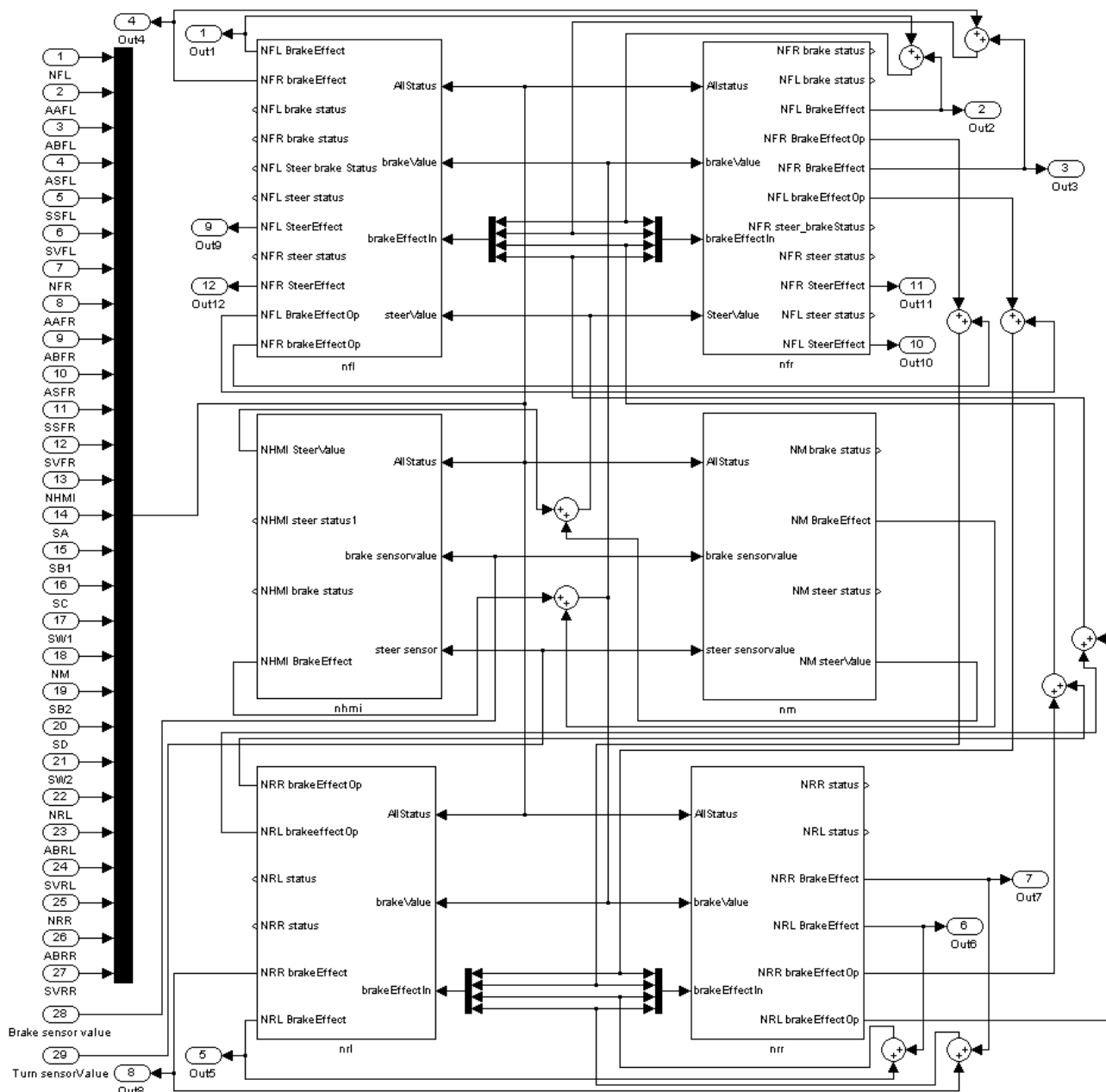
Aktuell \ Trasig	NFL	NFR
NFL	Avsluta applikation	Starta upp NFLs applikation på egen nod med samma status som egna applikationen
ASFL	Starta upp applikationen	Starta upp applikationen
ABFL	Ändra bromsverkan till 0%, behåll beräkningarna	Ändra bromsverkan till 0%, behåll beräkningarna
NFR	Starta upp NFRs applikation på egen nod med samma status som egna applikationen	Avsluta applikation
ASFR	Starta upp applikationen	Starta upp applikationen
ABFR	Ändra bromsverkan till 0%, behåll beräkningarna	Ändra bromsverkan till 0%, behåll beräkningarna
NHMI	-	-
SS1	-	-
SS2	-	-
NM	-	-

Tabell 2-4 – Ut signaler från MW

För Hjulenoderna:	För NHMI och NM:
Aktivsignalen (active)	Aktivsignalen (active)
0 = suspended	0 = suspended
1 = unsuspend	1 = unsuspend
Kontrollsignalen (cs)	Kontrollsignalen (cs)
0 = shut down (error)	0 = shut down (error)
2 = factor 0.3	
3 = factor 1	
4 = factor 0	
5 = factor 0.7	

2.4. Simulink-modell

Figur 2-18 visar en översikt av hela modellen där vi ser de sex noderna som block, alla insignaler till vänster och utsignalerna bredvid noderna.



Figur 2-18 – Översikt av Simulink-modellen

De insignaler som går in i en multiplexer (insignal 1-27) är statussignaler från samtliga komponenter i systemet, t.ex. så är insignal 1 statussignalen från NFL. Detta betyder att om insignal 1 har värdet 0 så betyder det att NFL inte längre är aktiv. De återstående insignalerna, dvs. insignal 28-29, är sensorvärden. Utsignalerna är värden till aktuatorerna. En mer noggrann beskrivning av dessa signaler visas i tabell 2-5.

Signal	Från/Till
In28	SB
In29	SW
Ut1	ABFL
Ut2	ABFL
Ut3	ABFR
Ut4	ABFR
Ut5	ABRL
Ut6	ABRL
Ut7	ABRR
Ut8	ABRR
Ut9	ASFL
Ut10	ASFL
Ut11	ASFR
Ut12	ASFR

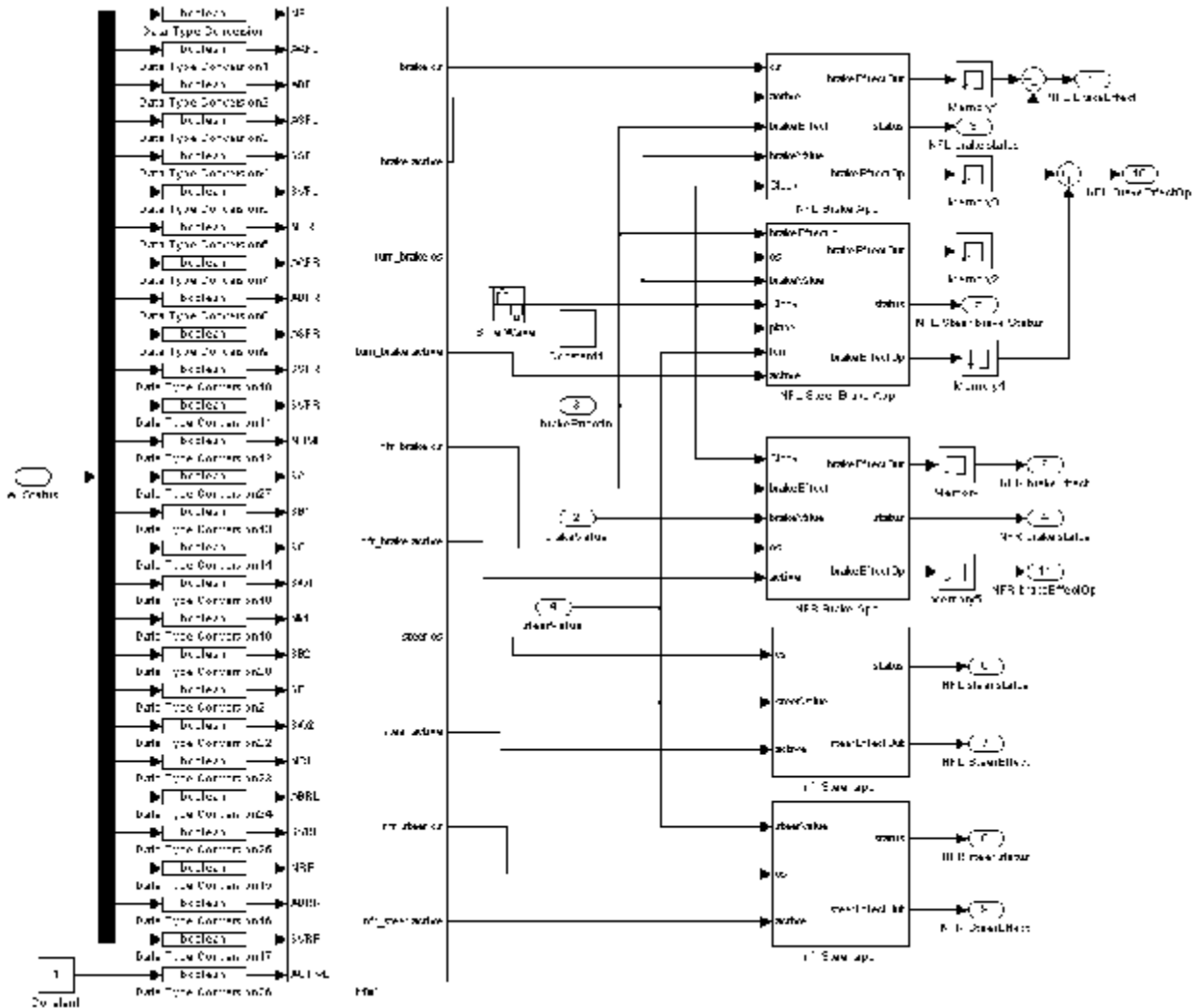
Tabell 2-5 – Insignaler till och utsignaler från modellen

Figur 2-19 visar NFL med dess applikationer. Blocket längst till vänster är MW och blocken till höger är applikationerna som körs på noden. Dessa applikationer är BBW, SBW och SBB samt två kopior av applikationerna BBW och SBW som finns på NFL för att möjliggöra migrering från NFR.

Den genererade sinussignalen används av noden som klocka.

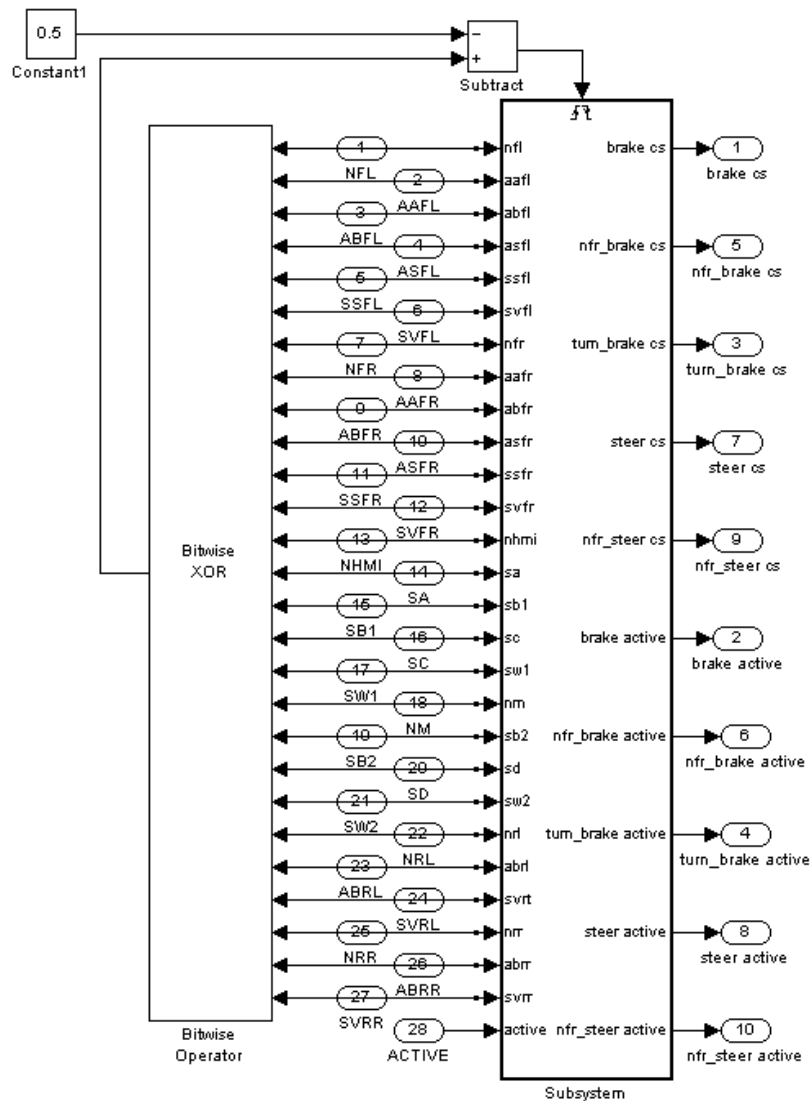
Vissa utsignaler måste, p.g.a. återkoppling, kopplas via minnesblock som fördröjer signalerna med en tidsenhet.

NFR är uppbyggd på precis samma sätt som NFL, med den enda skillnaden att värdet på konstanten som skickas in i "place"-ingången på SBB-applikationen är 2 istället för 1.



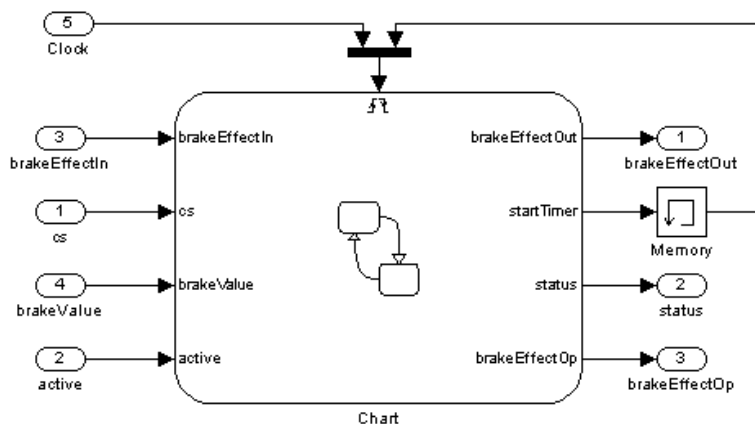
Figur 2-19 – Simulink-modell över NFL

Figur 2-20 visar hur MW-blocket i NFL är uppbyggt. I mitten ses insignalerna och på höger sida utsignalerna. I övrigt finns ett XOR-block, ett konstantvärde samt en adderare. De fungerar genom att ge en trigger-signal varje gång någon insignal ändras.



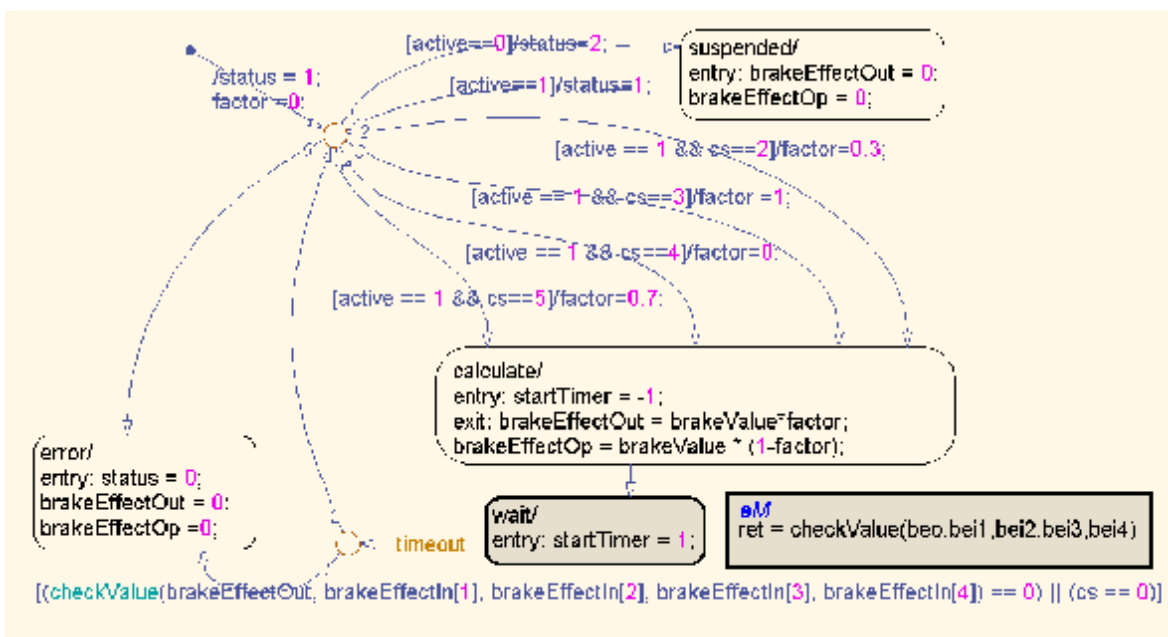
Figur 2-20 – Simulink-modell av MW i NFL

Bromsapplikationen visas i figur 2-21 med insignalerna till vänster och utsignalerna till höger i figuren. Återkoppling sker via en fördröjning för att vänta in resultat från övriga noder.



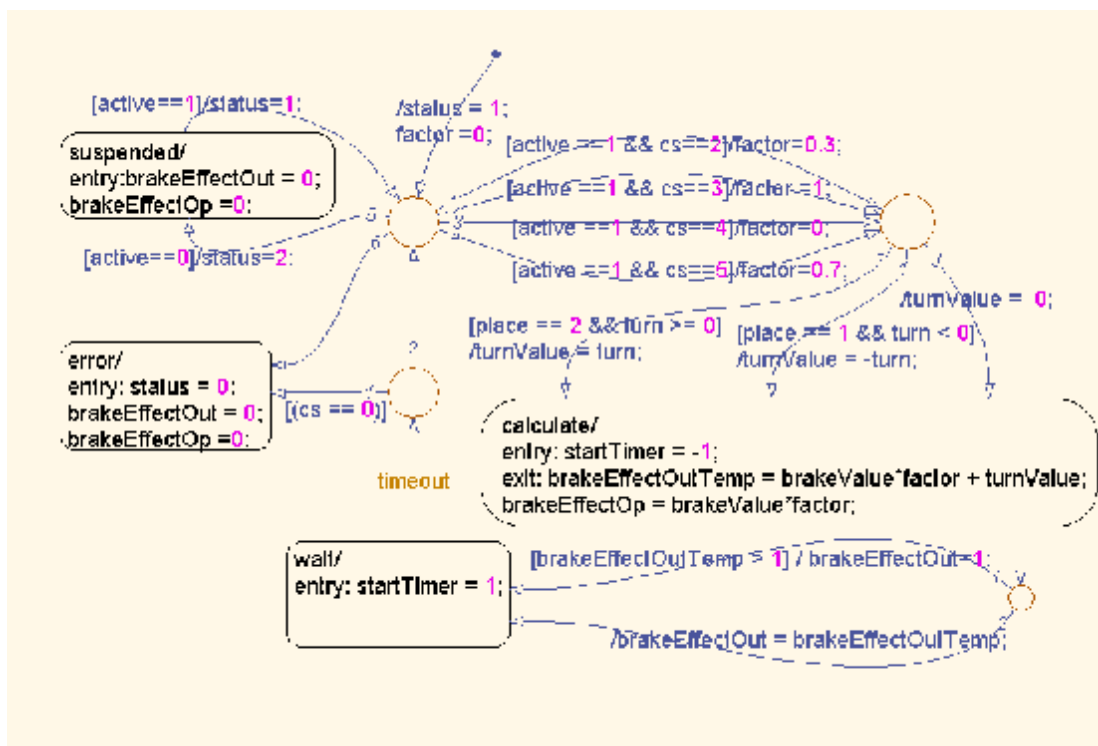
Figur 2-21 – Simulink-modell av BBW-applikationen

I figur 2-22 visas flödesschemat för hjulnodernas BBW-applikation. Ytterligare information om denna applikation finns under kapitel 2.3.1.



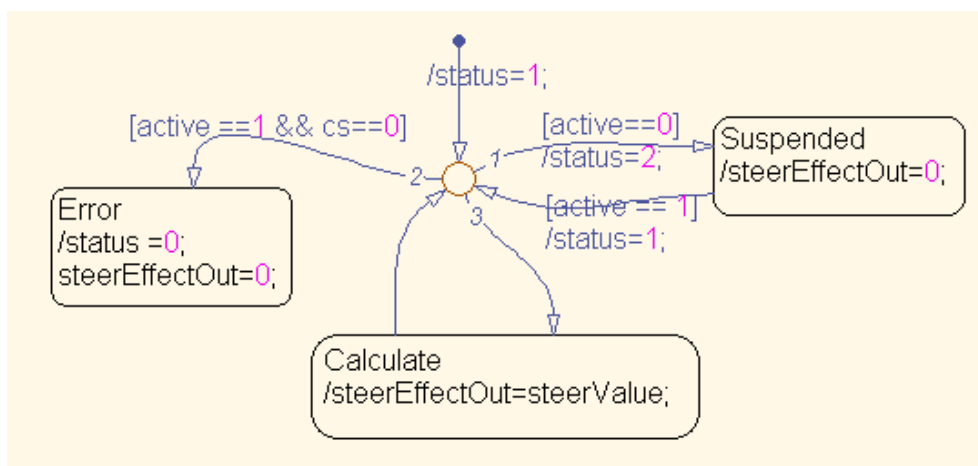
Figur 2-22 – Hjulnodernas BBW-applikation

Figur 2-23 visar flödesschemat för hjulnodernas SBB-applikation. Ytterligare information om denna applikation finns under kapitel 2.3.3.



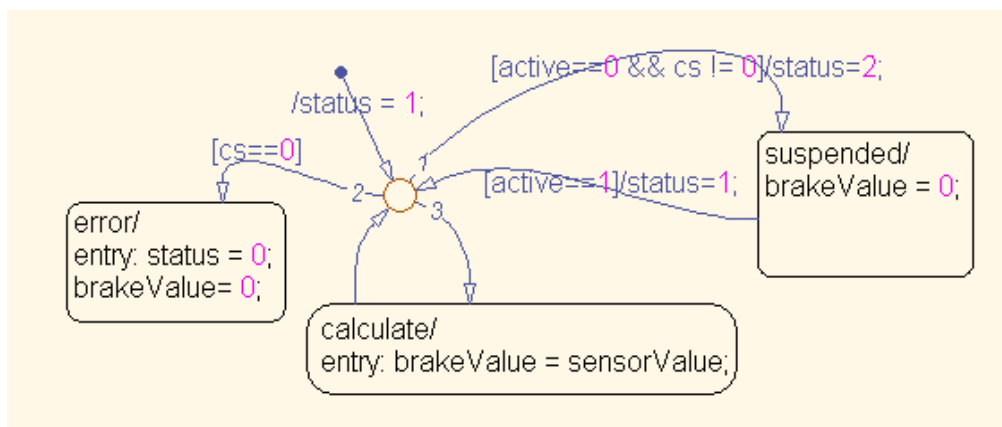
Figur 2-23 – Hjulnodernas SBB-applikation

Figur 2-24 visar flödesschemat för hjulnodernas SBW-applikation. Ytterligare information om denna applikation finns under kapitel 2.3.2.



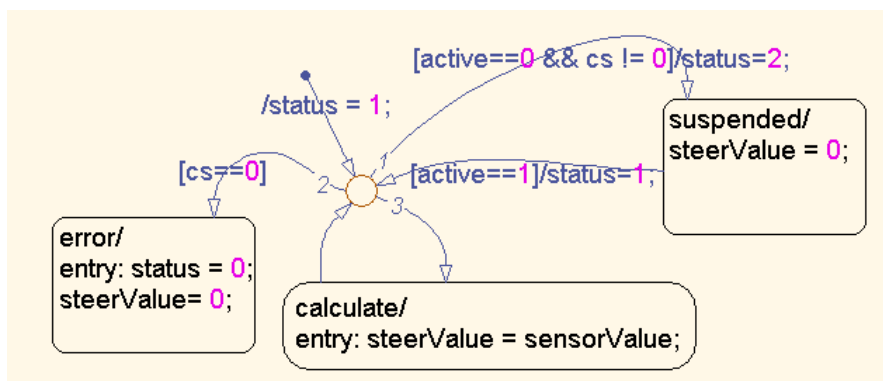
Figur 2-24 – Hjulnodernas SBW-applikationen

NM och NHMI är uppbyggda på samma sätt och har, utöver MW, bara två applikationer - broms och styrning, se figur 2-25. NM fungerar som en backupnod för NHMI och därför kommer inte någon applikation att köras på NM så länge NHMI fungerar. Av samma anledning kommer aldrig någon applikation att migreras från NM till NHMI. Sinussignalen och minnesblocket har samma funktioner som i NFL.



Figur 2-26 – BBW-applikationen på NHMI och NM

Figur 2-27 visar flödesschemat för NHMIs och NMs SBW-applikation. Ytterligare information om denna applikation finns under kapitel 2.3.2



Figur 2-27 – SBW-applikationen på NHMI och NM

På bakhjulsnoderna finns, förutom MW, bara bromsapplikationen, se figur 2-28. Att bromsapplikationen är den enda applikationen på dessa noder beror på att bakhjulen varken kan svänga eller driva fordonet. Därför finns på NRL en BBW-applikation samt en backup för NRRs BBW-applikation. Sinussignalen och minnesblocket har samma funktioner som i NFL.

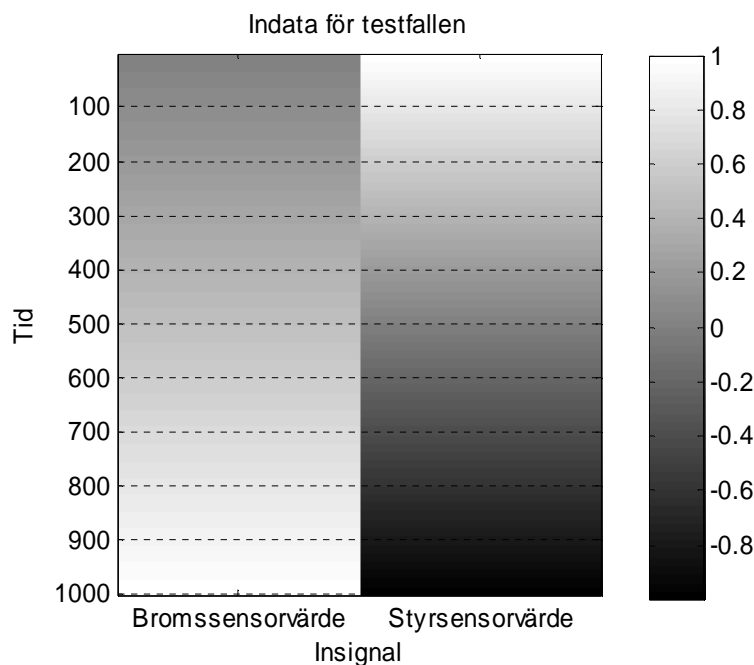
3. Simuleringsresultat

För att testa vår modell har vi valt skapa ett antal testfall som täcker in de fel som vår modell klarar av att hantera. De testfall, S1–S7, som vi har använt i våra tester av systemet visas i tabell 3-1. Förkortningarna som används för testerna förklaras i kapitel 6.3.

Tabell 3-1 Testfall

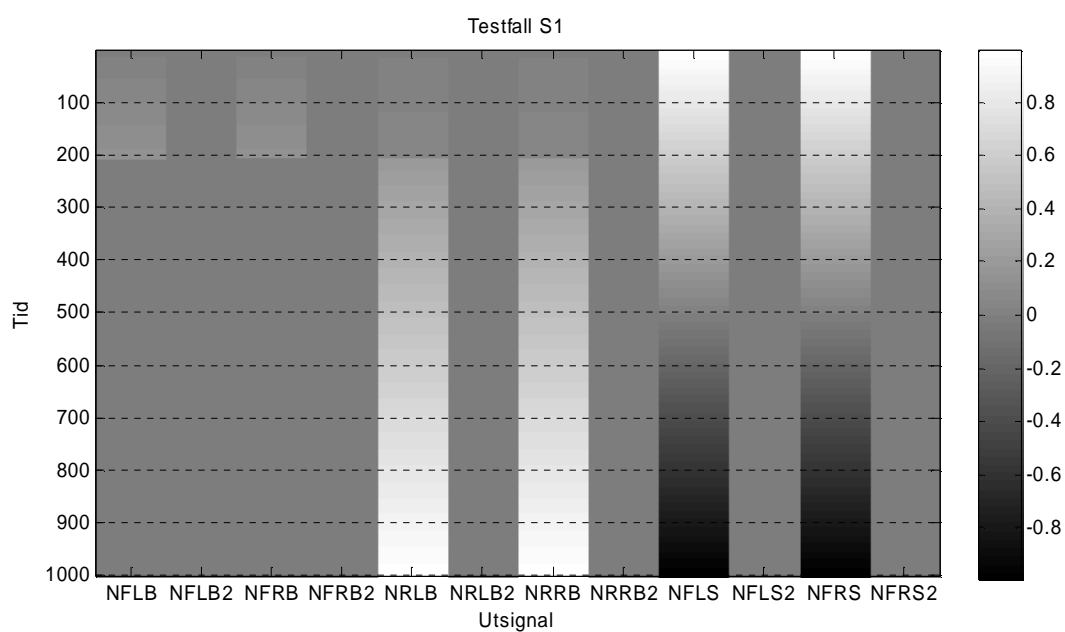
Testfall	Komponent som slutar fungera	Tid då felet inträffar
S1	ABFL	200
S2	NFR	200
S3	ABRL	200
S4	SW1	200
S5	NHMI	200
S6	SB1	200
S7	ASFR	200

I samtliga simuleringar ändras bromsvärdet linjärt från 0 till 1, dvs. från inget till max. Svängvärdet ändras från -1 till 1, vilket är max rattutslag åt båda håll, se figur 3-1.

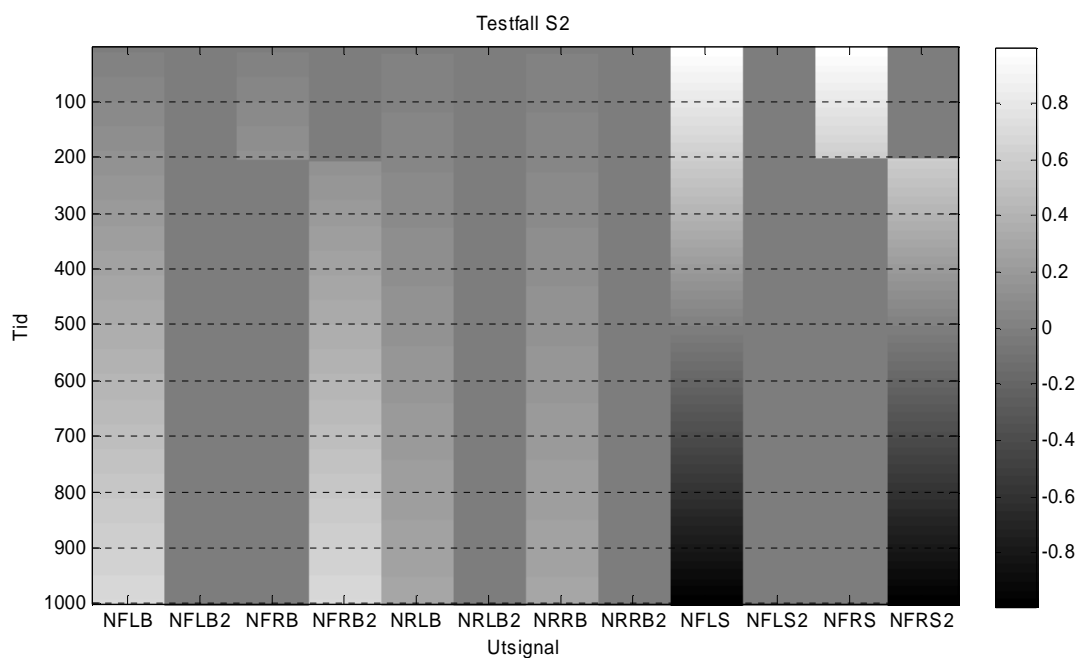


Figur 3-1 – Indata för testfallen

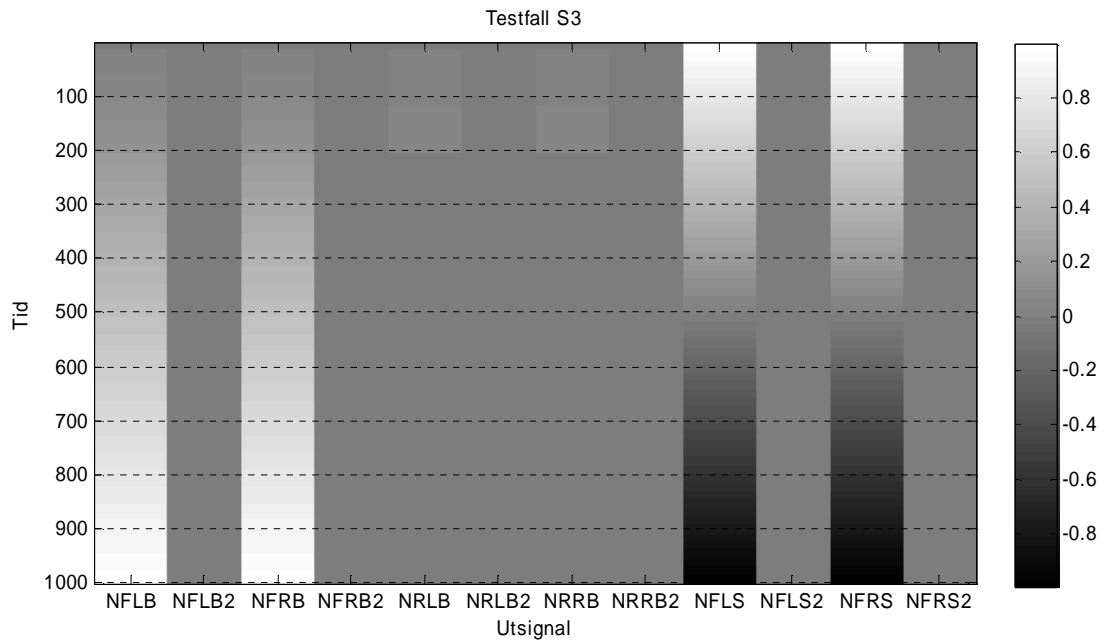
I figur 3-2 visas resultatet av vårt första testfall där bromsaktuatorn på vänster framhjul går sönder efter 200 tidsenheter. När detta sker stängs bromsapplikationerna av på båda framhjulsnoderna och bakhjulen ändras från att bara bromsa med 30 % till att överta hela bromsverkan.

**Figur 3-2 – Testfall S1**

I figur 3-3 går höger framhjulsnod sönder och vi ser att funktionen för både styrning och broms tas över av motsvarande applikation på vänster framhjulsnod istället. Applikationerna på de övriga noderna påverkas inte.

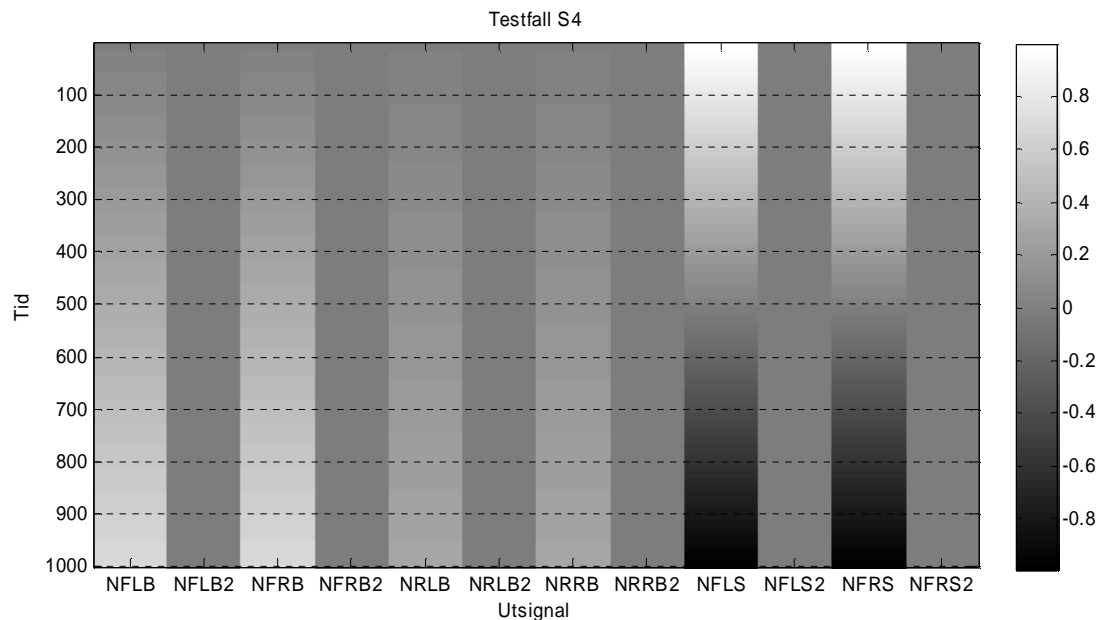
**Figur 3-3 – Testfall S2**

Från figur 3-4 slutar bromsaktuatorn på höger bakhjul att fungera. Man ser då att båda bakhjulen slutar att bromsa och att framhjulen ändras från att bromsa med 70 % verkan till att överta hela bromsverkan.



Figur 3-4 – Testfall S3

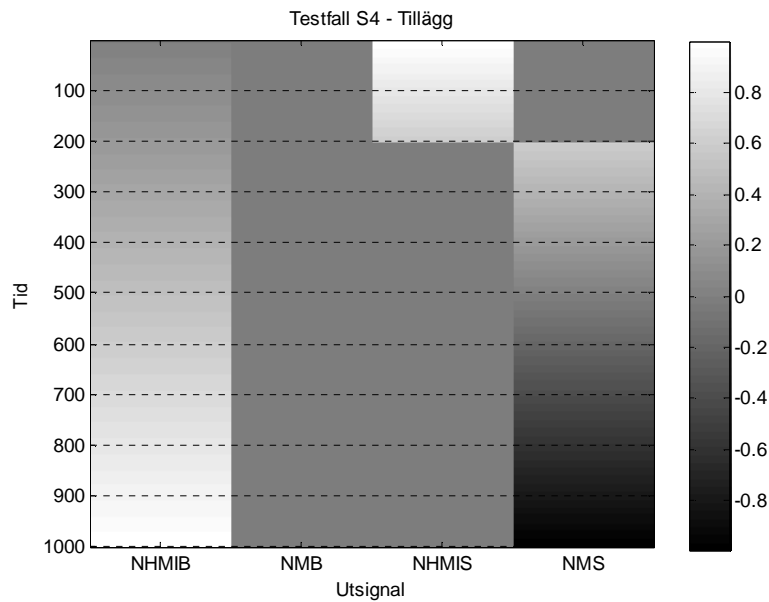
I figur 3-5 visas resultatet av scenario 4. Vi kan inte från detta diagram se att något är fel, men i detta testfall går rattsensorn sönder och dess funktion startas på NM.



Figur 3-5 – Testfall S4

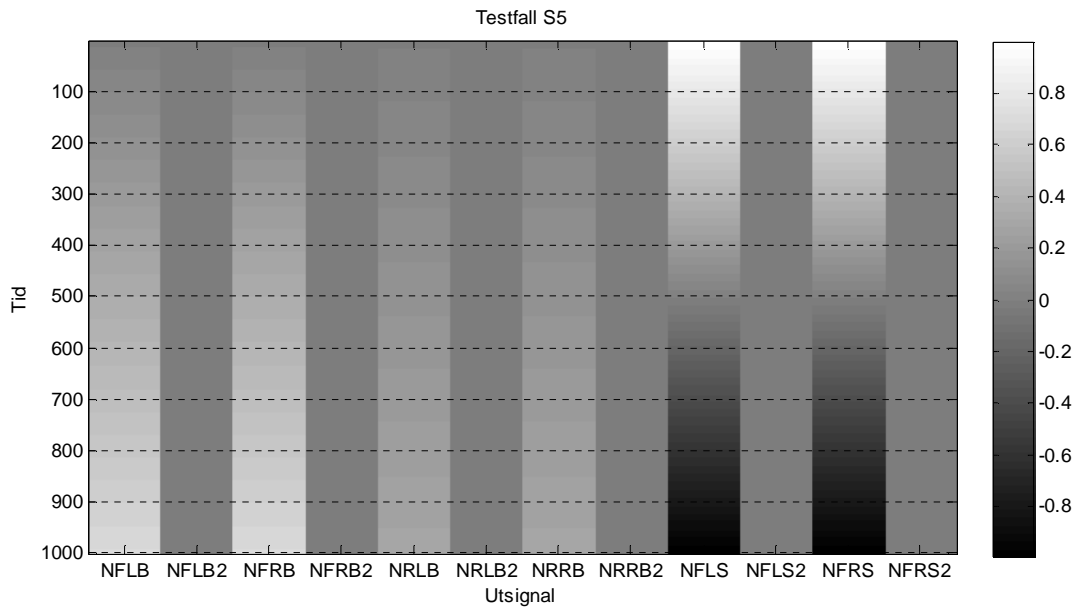
För att tydligare visa vad som händer i testfall 4 visas i figur 3-6 samma testfall fast med fokus på andra signaler. Det syns nu tydligt att sensorn slutar fungera och att NHMI därför inte

längre ger någon utsignal för styrningen. Den signalen kommer nu istället från NM.



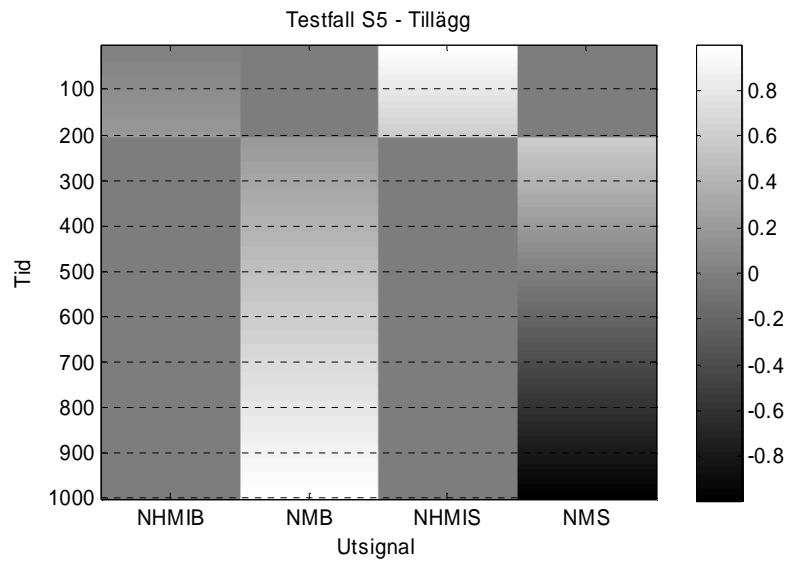
Figur 3-6 – Tillägg för testfall S4

Scenario 5 i figur 3-7 visar full funktionalitet och vi ser att resultatet är identiskt med figur 3-5, vilket det också borde vara.



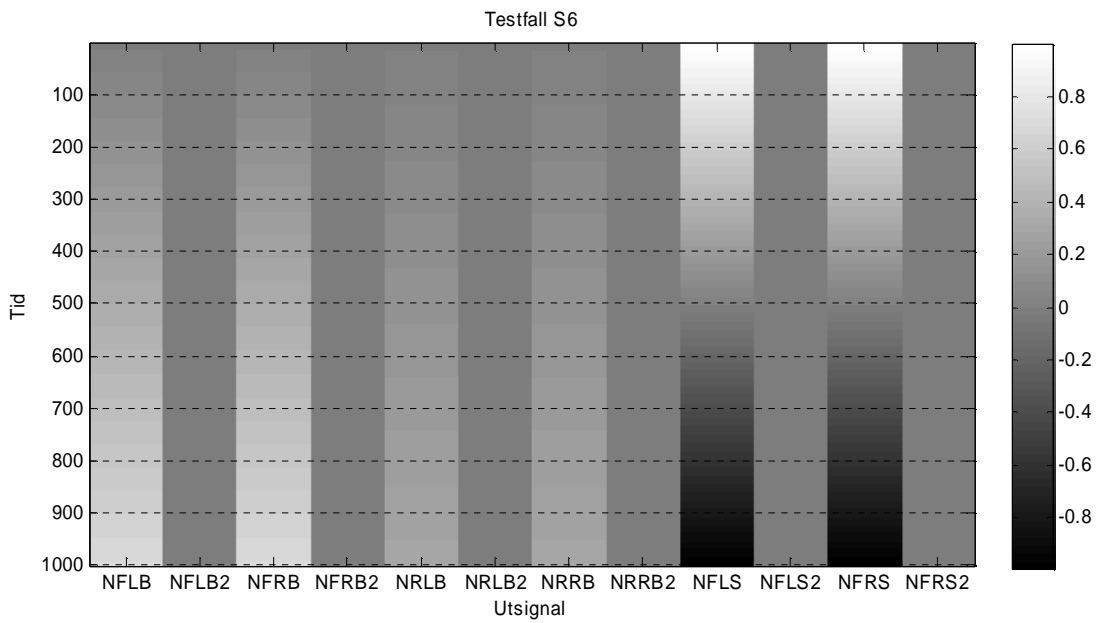
Figur 3-7 – Testfall S5

Skillnaden jämfört med föregående scenario ligger i att NHMI i detta testfall slutar fungera och alla dess applikationer migrerar då till NM, se figur 3-8.

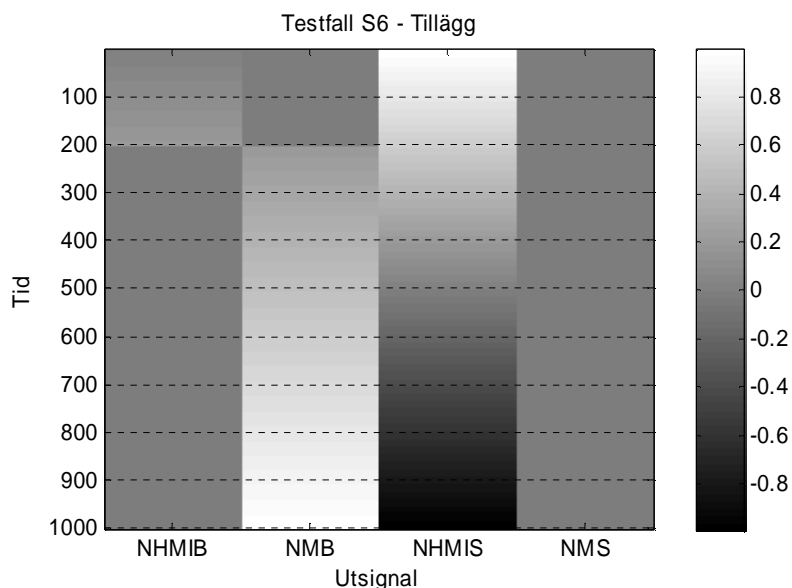


Figur 3-8 – Tillägg för testfall S5

I scenario 6 är det bromssensorn SB1 som slutar fungera och bromsapplikationen migreras då från NHMI till NM som har en redundant sensor. Resultatet visas i figur 3-9 och figur 3-10.



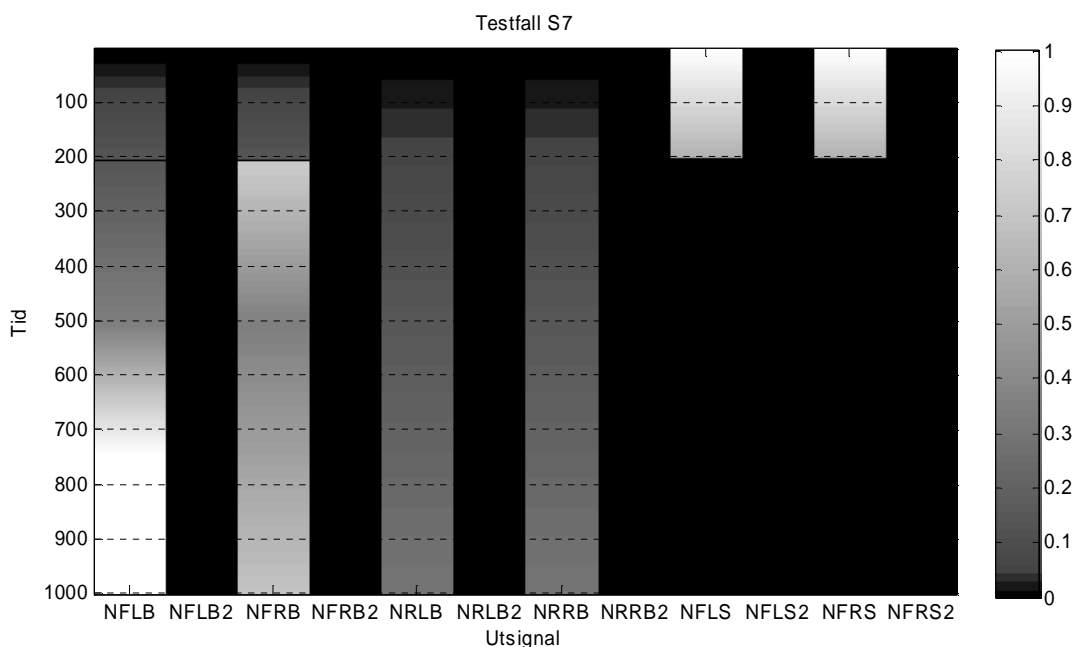
Figur 3-9 – Testfall S6



Figur 3-10 – Tillägg för testfall S6

I figur 3-11 visas resultatet från testfall 7. När A_{SFR} går sönder slutar båda hjulen att svänga. Eftersom styrvärdet vid detta tillfälle var positivt kommer bromsverkan på höger framhjul att superpositioneras med styrvärdet. När styrvärdet senare blir negativt kommer styrvärdet istället superpositioneras med bromsverkan på vänster framhjul. Bakhjulen berörs inte av detta.

Färgskalan i figur 3-11 är ändrad eftersom det inte förekommer några negativa värden i det här testfallet. Detta beror på att det negativa styrvärdet som skulle ha fått fordonet att svänga åt vänster istället ändrats till en positiv bromsverkan på vänster framhjul.



Figur 3-11 – Testfall S7

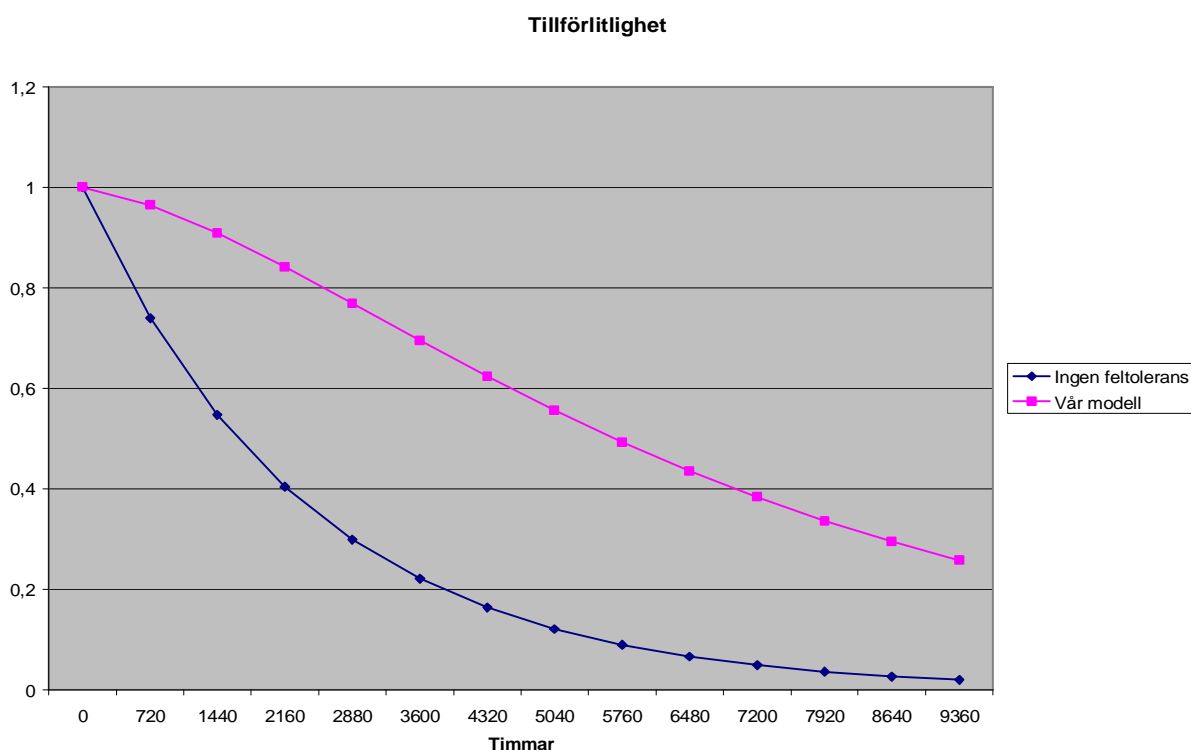
Felintensitet per timme och MTTF (Mean Time To Failure) för hela modellen:

	Felintensitet/h	MTTF (h)
Ingen feltolerans	$4,19 \cdot 10^{-4}$	2390
Vår modell	$1,42 \cdot 10^{-4}$	7020

Vi har förenklat modellen genom att inte göra några beräkningar på felintensitet när fordonet ej används så om resultaten tycks vara nedslående får man tänka på att det endast är beräknat på när fordonet är i drift, se figur 4-2. Om vi antar att fordonet används 2 timmar om dagen kommer MTTF att motsvara:

Ingen feltolerans: $2\,390 \cdot 12 = 28\,700$ timmar $\approx 3,3$ år.

Vår modell: $7\,024 \cdot 12 = 84\,300$ timmar $\approx 9,6$ år.



Figur 4-2 - Tillförlitlighetsdiagram

Det finns naturligtvis system med större tillförlitlighet än det vi valde. Problemet är att om man skulle ha till exempel ett TMR-system (Triple Modular Redundancy) för alla elektriska komponenter i bilen så skulle den bli så tung och dyr att ingen vill betala för tillförlitligheten. Den valda modellen var istället ett försök att skapa ett feltolerant system till en rimlig kostnad där vi enkelt kan simulera våra applikationsbyten. Aktuatorerna är den svagaste länken i kedjan med en betydligt högre felintensitet än sensorer och datornoder och har därför den största förbättringspotentialen.

4.2. Problem

Under arbetet med att utveckla modellerna har en del problem stötts på. Det enskilt största problemet framkom några veckor in i projektet. Vi upptäckte då att uppgiften var underspecificerad och att vi därför hade arbetat fram en lösning som innebar att funktionen som numera finns i MW var inbyggt i applikationen. Att bygga upp applikationerna på så sätt ledde till att modellerna blev onödigt komplexa. Detta problem löstes genom en del samtal och till slut var uppgiften tillräckligt välspecificerad för att vi skulle kunna fortsätta jobba.

Ett önskemål under utvecklingen av modellerna var att vi skulle kunna använda oss av "model referencing", dvs. kunna återanvända våra applikationsmodeller istället för att ha kopior av dem i noderna. För att överhuvudtaget få referenserna att fungera var vi tvungna att byta till en senare version av Matlab. Att ha referenser skulle kunna underlätta arbetet väsentligt, men tyvärr ledde det till att debuggningsarbetet i Simulink blev näst intill omöjligt på grund av intetsägande felmeddelanden. Matlab gav t.ex. ett "non-structural array"-felmeddelande utan någon ytterligare hänvisning, vilket senare visade sig bero på att det saknades invärden till vissa funktioner i startögonblicket.

Vi hade också problem med att skapa funktionen som kontrollerar om beräknad bromsverkan är korrekt. Flera alternativ testades och det bästa verkade vara att kalla på en vanlig m-fil (en fil innehållande matlabkod) från stateflow-diagrammet. Det var problem med detta också, eftersom funktionen inte gav något svar om man inte anropade den på rätt sätt. Vi fick i ett senare skede byta till en s.k. inbäddad Matlab-funktion (funktionen är inkluderad i modellen) istället eftersom kodgenereringen annars inte fungerade. Indata till funktionen gjordes i första läget olika för de olika noderna vilket resulterade i att scenariona inte stämde och vi fick inkonsistens i systemet. Vi har därför valt att skicka in samtliga invärden, även egna värdet, för att alla noderna skall hamna i samma tillstånd. Vi fick dock ett litet avrundningsfel vid beräkning i noderna, så vi nöjde oss med att säga att resultaten var lika om skillnaden var mindre än 10^{-10} .

För vår SBB-applikation kunde inte den existerande BBW-applikationen användas, eftersom eventuell migrering i så fall blir allt för komplicerad.

Vi tyckte ett tag att applikationerna i Simulink skulle avslutas när någonting blev fel. Som det är nu så går de bara in i ett "Error"-tillstånd. Men operativsystemet kommer ändå att köra de program som är igång och inte ge någon processortid till de applikationer som inte skall köras. Vilka applikationer som skall köras eller inte köras kommer operativsystemet att få reda på genom våra ut signaler från MW.

Det uppstod problem i bromsapplikationerna på bakhjulen eftersom faktorn där sattes till 0,7 vid start (precis som det skall vara på framhjulen) så kom vi fram till att det måste finnas två invärden till samtliga våra applikationer, en kontrollsignal och en aktiv. Kontrollsignalen skall vara den som exempelvis avgör vad faktorn skall vara i bromsapplikationen, denna signal skickas

direkt till applikationen. Aktiv-signalen skickas till realtidsoperativsystemet som med dess hjälp avgör om applikationen skall ha processortid eller helt och hållet avslutas.

Samtliga applikationer behöver ha en styrsignal, om inte annat så för att gå till "Error"-tillståndet för att applikationen skall avslutas ordentligt.

5. Slutsats

Vårt mål med arbetet var att skapa applikationerna för framtida experiment inom området för partitionerade system där redundanshantering och task migration är en del. Vi har skapat en ny fungerande redundant modell som kommer att kunna användas för fortsatta experiment och utveckling. Modellen har byggts så att kodgenerering från Real-Time Workshop (del av Simulink för kodgenerering) är möjlig. Men det krävs fortfarande en del insatser för att det skall bli direkt körbart på en GAST-nod, exempelvis så måste den genererade koden delas upp så att varje applikation blir en egen fil.

Det har implementerats tre olika applikationer och modellen är uppbyggd av moduler för att göra det enkelt att utöka antalet applikationer och noder efter framtida behov.

Vi har även visat att vår modell är en rimlig kompromiss mellan feltolerans och komplexitet samt att den har fördelar jämfört med en icke feltolerant modell.

Det är svårt att mäta i modellen hur många klockcykler en applikation kommer att kräva när det är implementerat i en specifik miljö och det hade därför varit av intresse att kunna köra applikationerna på en riktig nod för att få empiriska resultat.

5.1. Framtida arbete

Det finns några saker som eventuellt kan fortsättas med om man vill driva projektet vidare. Den första handlar om hur man skall hantera operativsystemet och hur schemalaggningsen skall fungera. Då bör en trigger-signal läggas till på varje applikation för att göra det möjligt för ett realtidsoperativsystem att schemalägga våra applikationer. Detta beror på hur operativsystemet hanterar applikationerna och kommer att behöva anpassas efter det.

För att automatgenerering skall fungera utan modifikation av koden krävs bl.a. externa funktionsanrop. Även hur man skall generera kod individuellt för de olika applikationerna är ett problem. Helst skulle man ha en modell av Flexray-bussarna i Simulink. Det kommer att startas ett examensarbete inom GAST-projektet som kommer att behandla just kodgenerering från Simulink.

För att fullständigt kunna använda task migration krävs att systemet har en fungerande membership agreement-funktion. Det vore också en fördel om samtliga noder har kontakt med alla aktuatorer och sensorer i modellen. För att optimera resursfördelningen bör task migration ske dynamiskt.

Vid fel i applikationerna bör man testa om felet endast är transienta och i så fall hantera det på lämpligt sätt.

När systemet befinner sig i degraderat tillstånd så vore det lämpligt att begränsa den maximala hastigheten på fordonet.

6. Förkortningar

6.1. Komponenter

S_A	-	Accelerator (gaspedal)
S_B	-	Brake (bromspedal)
S_C	-	Cruise Control (farthållare av/på)
S_D	-	Drifting (sladdsensor, förflyttning i sidled)
S_S	-	Steering (vridning på hjulet)
S_V	-	Velocity (hastigheten på hjulet)
S_W	-	Steering Wheel (ratt)
A_A	-	Acceleration (drivningen på hjulen)
A_B	-	Brake (bromsar hjulen)
A_S	-	Steering (vrider hjulen)
NM	-	Main Node (huvudnoden)
NHMI	-	Human Machine Interface Node (förarmiljönoden)
NFL	-	Front Left Wheel Node (vänster framhjulsnod)
NFR	-	Front Right Wheel Node (höger framhjulsnod)
NRL	-	Rear Left Wheel Node (vänster bakhjulsnod)
NRR	-	Rear Right Wheel Node (höger bakhjulsnod)
NW	-	Any Wheel Node (hjulsnod)
NFW	-	Any Front Wheel Node (framhjulsnod)
NRW	-	Any Rear Wheel Node (bakhjulsnod)

6.2. Applikationer

SBB	-	Steer By Brake
BBW	-	Brake By Wire
SBW	-	Steer By Wire
MW	-	MiddleWare

6.3. Testfall

NFLB	-	bromssignalen som går till A_{BFL} från NFL
NFLB2	-	backupbromssignalen till A_{BFL} från NFR.
NFRB	-	bromssignalen som går till A_{BFR} från NFR
NFRB2	-	backupbromssignalen till A_{BFR} från NFL.
NRLB	-	bromssignalen som går till A_{BRL} från NRL
NRLB2	-	backupbromssignalen till A_{BRL} från NRR.
NRRB	-	bromssignalen som går till A_{BRR} från NRR.
NRRB2	-	backupbromssignalen till A_{BRR} från NRL
NFLS	-	styrsignalen som går till A_{SFL} från NFL
NFLS2	-	backupsignalen som går till A_{SFL} från NFR.
NFRS	-	styrsignalen som går till A_{SFR} från NFR
NFRS2	-	backupsignalen som går till A_{SFR} från NFL.

7. Referenser

- [1] Bergenhem, Carl, Dependable Embedded Components and Systems,
http://www.sp.se/ENCRESS/sv/Material_07_nov_2005/DECOS%20dissemination%20ENCRESS%2005%2005.pdf, 2006-01-04.
- [2] Ericsson, Henric, The DECOS Integrated Diagnostic Architecture,
http://www.sp.se/ENCRESS/sv/Material_07_nov_2005/DECOS-Diagnostic-Architecture_handout.pdf, 2006-01-04.
- [3] Forsberg, Kristina, Design Principles of Fly-By-Wire Architectures, Chalmers reproservice, 2003.
- [4] Källvik, Magnus och Eriksson, Jonas, Sensorer och aktuatorer för G1 och G2, lämpade för en bromsapplikation, pågående examensarbete 2006-01-08.
- [5] Sivencrona, Håkan, Tidsstyrda distribuerade kommunikationssystem, Elektronik i Norden, nr 11, pp 32, 2002,
<http://img.cmpnet.com/edtn/europe/elektronik/pdf/2002/11sid32.pdf>, 2006-01-07
- [6] ENCRESS-seminarium, Borås, 051007
- [7] Flexray, <http://www.flexray.com>, 2006-01-04.
- [8] GAST-projektet, <http://www.chl.chalmers.se/gast/>, 2006-01-04.
- [9] SCADE: Modelleringsprogram från Esterel-Technologies,
<http://www.esterel-technologies.com/products/scade-suite/overview.html>, 2006-01-04.
- [10] Simulink: Tilläggsprogram för Matlab av MathWorks,
<http://www.mathworks.com/products/simulink/>, 2006-01-04.
- [11] X-By-Wire Overview
http://www.sp.se/ENCRESS/sv/Material_7_dec_2001/XbW%20ENCRESS.pdf, 2006-01-04.